PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-353678
(43)Date of publication of application : 25.12.2001

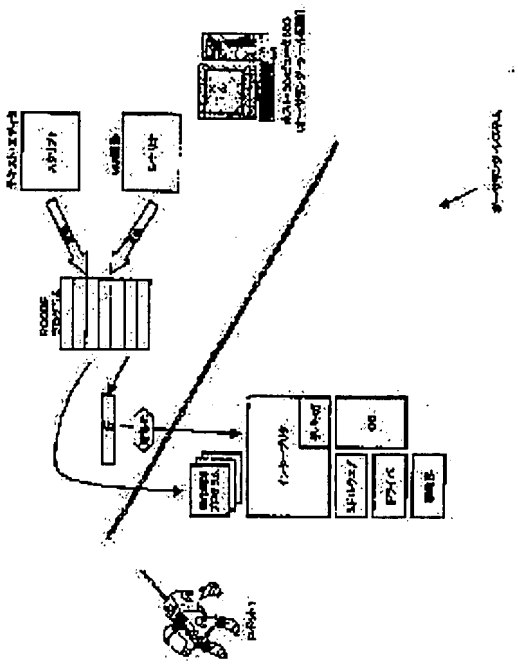
(51)Int.Cl. B25J 13/00
B25J 5/00
B25J 9/16
B25J 9/22
B25J 19/00

(21)Application number : 2000-175159
(22)Date of filing : 12.06.2000

(71)Applicant : SONY CORP
(72)Inventor : KASUGA TOMOAKI
OKITA FUMIKO

(54) AUTHORING SYSTEM AND METHOD AND STORAGE MEDIUM

(57)Abstract:
PROBLEM TO BE SOLVED: To originate an operating pattern by use of an assembly of parts which specify the operating condition of a robot and a process routine.
SOLUTION: A user originates and edits a specified scenario of a robot 1 through a mouse operation using a GUI screen. An authoring tool converts the scenario originated and edited into a mnemonic mode called RCODE. During debugging of an RCODE operation control program, an RCODE program is taken out row by row and enciphered for transfer to the robot by use of a wireless communications means. At the robot, an interpreter interprets and executes the RCODE and performs debugging.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2001－353678
（P2001－353678A）
(43)公開日　平成13年12月25日(2001.12.25)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｂ２５Ｊ　13/00		Ｂ２５Ｊ　13/00	Ｚ	３Ｆ０５９
5/00		5/00	Ｃ	３Ｆ０６０
9/16		9/16		
9/22		9/22	Ｚ	
19/00		19/00	Ｋ	

審査請求　未請求　請求項の数29　ＯＬ　（全 34 頁）

(21)出願番号　特願2000－175159(P2000－175159)

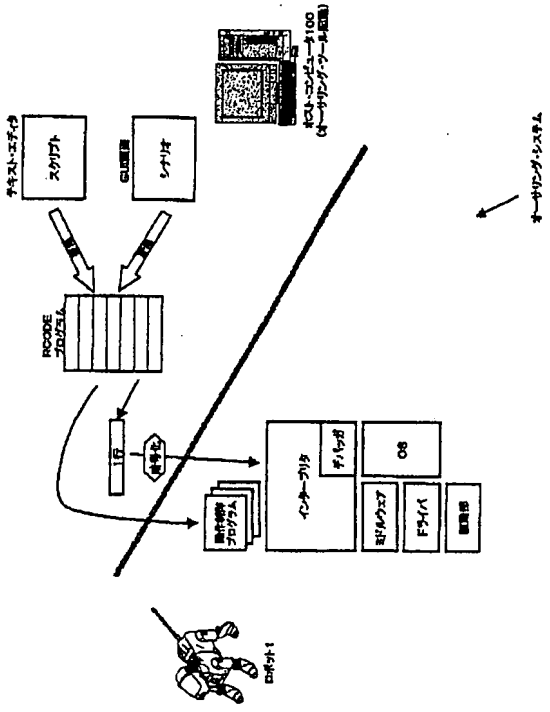
(22)出願日　平成12年６月12日(2000.6.12)

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号
(72)発明者　春日　知昭
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(72)発明者　沖田　文子
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(74)代理人　100101801
弁理士　山田　英治　（外２名）

最終頁に続く

(54)【発明の名称】　オーサリング・システム及びオーサリング方法、並びに記憶媒体

(57)【要約】
【課題】　ロボットの動作状態や処理ルーチンを規定する部品の集合を用いて動作パターンを作成する。
【解決手段】　ユーザは、ＧＵＩ画面を用いてマウス操作によりロボット１の規定のシナリオを作成・編集する。オーサリング・ツールは、作成・編集したシナリオをＲＣＯＤＥと呼ばれるニーモニック・コードに変換する。ＲＣＯＤＥ動作制御プログラムのデバッグ時には、ＲＣＯＤＥプログラムを１行ごとに取り出して、暗号化し無線通信手段を利用してロボット側に逐次転送する。ロボット側ではＲＣＯＤＥをインタープリタが逐次解釈・実行してデバッグ処理する。



1

【特許請求の範囲】

【請求項１】ロボットの動作制御プログラムを作成･編集するためのオーサリング･システムであって、

ユーザに対して該ロボットの動作シナリオを作成・編集するための作業場を提供する編集部と、

前記編集部を介して作成･編集されたシナリオを前記ロボット上で解釈可能なプログラム･コードに変換する変換部と、を具備することを特徴とするオーサリング･システム。

【請求項２】前記編集部は、頻繁に使用する動作状態や処理ルーチン又はその雛型を部品化して用意する部品群と、座標指示装置を介した指示操作により前記部品群の各部品を取捨選択して配置する操作画面とをユーザに提供することを特徴とする請求項１に記載のオーサリング･システム。

【請求項３】前記編集部は、所定のプログラム言語形式のスクリプトを作成･編集する環境をユーザに提供することを特徴とする請求項１に記載のオーサリング･システム。

【請求項４】前記変換部は、部品の配置により表現された動作シナリオ及び／又はスクリプト形式で記述された動作シナリオを、前記ロボットにおいて解釈可能なニーモニック･コードに変換することを特徴とする請求項１に記載のオーサリング･システム。

【請求項５】さらに、前記変換部において変換されたプログラム･コードを前記ロボットに転送するための通信手段を備えることを特徴とする請求項１に記載のオーサリング・システム。

【請求項６】前記通信手段は前記ロボットと無線データ通信を行うことを特徴とする請求項５に記載のオーサリング･システム。

【請求項７】前記通信手段は、前記変換部において変換されたプログラム･コードを１ステップずつ取り出して前記ロボットに転送することを特徴とする請求項５に記載のオーサリング･システム。

【請求項８】前記通信手段は、前記変換部において変換されたプログラム･コードを暗号化して前記ロボットに転送することを特徴とする請求項５に記載のオーサリング･システム。

【請求項９】さらに、

前記通信手段を介して受信したプログラム･コードを解釈する解釈部と、

前記解釈部による解釈結果に従って前記ロボットを駆動する駆動制御部と、を前記ロボット上に備えることを特徴とする請求項５に記載のオーサリング･システム。

【請求項１０】前記解釈部は、受信したプログラム･コードを１ステップ単位で解釈して実行することを特徴とする請求項９に記載のオーサリング･システム。

【請求項１１】前記ロボットは、現在の姿勢から直接遷移可能な動作や姿勢並びにある動作や姿勢を経由してな

2

ら遷移可能となる姿勢などに関する姿勢遷移制限情報を有し、

前記駆動制御部は、該姿勢遷移制限情報を基に、プログラム･コードの指示内容を姿勢遷移可能な形式に変換する、ことを特徴とする請求項９に記載のオーサリング･システム。

【請求項１２】前記姿勢遷移制限情報は、前記ロボットがとり得る姿勢を示すノードと、遷移可能な２つのノード間を結ぶ動作アークで構成される有向グラフ形式で保持され、

前記駆動制御部は、該有向グラフを探索してプログラム･コードの指示内容を姿勢遷移可能な形式に変換する、ことを特徴とする請求項１１に記載のオーサリング･システム。

【請求項１３】ロボットの動作制御プログラムを作成･編集するためのオーサリング方法であって、

ユーザに対して該ロボットの動作シナリオを作成・編集するための作業場を提供する編集ステップと、

前記編集ステップを介して作成･編集されたシナリオを前記ロボット上で解釈可能なプログラム･コードに変換する変換ステップと、を具備することを特徴とするオーサリング方法。

【請求項１４】前記編集ステップでは、頻繁に使用する動作状態や処理ルーチン又はその雛型を部品化して用意する部品群と、座標指示装置を介した指示操作により前記部品群の各部品を取捨選択して配置する操作画面とをユーザに提供することを特徴とする請求項１３に記載のオーサリング方法。

【請求項１５】前記編集ステップでは、所定のプログラム言語形式のスクリプトを作成･編集する環境をユーザに提供することを特徴とする請求項１３に記載のオーサリング方法。

【請求項１６】前記変換ステップでは、部品の配置により表現された動作シナリオ及び／又はスクリプト形式で記述された動作シナリオを、前記ロボットにおいて解釈可能なニーモニック･コードに変換することを特徴とする請求項１３に記載のオーサリング方法。

【請求項１７】さらに、前記変換ステップにおいて変換されたプログラム･コードを前記ロボットに転送するための通信ステップを備えることを特徴とする請求項１３に記載のオーサリング方法。

【請求項１８】前記通信ステップでは前記ロボットと無線データ通信を行うことを特徴とする請求項１７に記載のオーサリング方法。

【請求項１９】前記通信ステップは、前記変換ステップにおいて変換されたプログラム･コードを１ステップずつ取り出して前記ロボットに転送することを特徴とする請求項１７に記載のオーサリング方法。

【請求項２０】前記通信ステップでは、前記変換部において変換されたプログラム･コードを暗号化して前記ロ

ボットに転送することを特徴とする請求項１７に記載のオーサリング方法。

【請求項２１】さらに、
前記通信ステップを介して受信したプログラム･コードを解釈する解釈ステップと、
前記解釈ステップによる解釈結果に従って前記ロボットを駆動する駆動制御ステップと、を前記ロボット上で実行することを特徴とする請求項１７に記載のオーサリング方法。

【請求項２２】前記解釈ステップでは、受信したプログラム･コードを１ステップ単位で解釈して実行することを特徴とする請求項２１に記載のオーサリング方法。

【請求項２３】前記ロボットは、現在の姿勢から直接遷移可能な動作や姿勢並びにある動作や姿勢を経由してなら遷移可能となる姿勢などに関する姿勢遷移制限情報を有し、
前記駆動制御ステップでは、該姿勢遷移制限情報を基に、プログラム･コードの指示内容を姿勢遷移可能な形式に変換する、ことを特徴とする請求項２１に記載のオーサリング方法。

【請求項２４】前記姿勢遷移制限情報は、前記ロボットがとり得る姿勢を示すノードと、遷移可能な２つのノード間を結ぶ動作アークで構成される有向グラフ形式で保持され、
前記駆動制御ステップでは、該有向グラフを探索してプログラム･コードの指示内容を姿勢遷移可能な形式に変換する、ことを特徴とする請求項２３に記載のオーサリング方法。

【請求項２５】ロボットの動作制御プログラムを作成･編集するためのオーサリング処理をコンピュータ･システム上で実行するように記述されたコンピュータ･ソフトウェアをコンピュータ可読形式で物理的に格納した記憶媒体であって、前記コンピュータ･ソフトウェアは、
ユーザに対して該ロボットの動作シナリオを作成・編集するための作業場を提供する編集ステップと、
前記編集ステップを介して作成･編集されたシナリオを前記ロボット上で解釈可能なプログラム･コードに変換する変換ステップと、を具備することを特徴とする記憶媒体。

【請求項２６】前記編集ステップでは、頻繁に使用する動作状態や処理ルーチン又はその雛型を部品化して用意する部品群と、座標指示装置を介した指示操作により前記部品群の各部品を取捨選択して配置する操作画面とをユーザに提供することを特徴とする請求項２５に記載の記憶媒体。

【請求項２７】前記編集ステップでは、所定のプログラム言語形式のスクリプトを作成･編集する環境をユーザに提供することを特徴とする請求項２５に記載の記憶媒体。

【請求項２８】前記変換ステップでは、部品の配置により表現された動作シナリオ及び／又はスクリプト形式で記述された動作シナリオを、前記ロボットにおいて解釈可能なニーモニック･コードに変換することを特徴とする請求項２５に記載の記憶媒体。

【請求項２９】さらに、前記変換ステップにおいて変換されたプログラム･コードを前記ロボットに転送するための通信ステップを備えることを特徴とする請求項２５に記載の記憶媒体。

【発明の詳細な説明】

【０００１】
【発明の属する技術分野】本発明は、所定のシナリオに従ったデータを作成するためのオーサリング・システム及び方法に係り、特に、ロボットの所定の動作パターンを記述する一連のコマンド／データを作成するオーサリング・システム及び方法に関する。

【０００２】更に詳しくは、本発明は、ロボットの動作状態を規定する部品の集合を用いて動作パターンを作成するオーサリング・システム及び方法に係り、特に、各部品をコンピュータ・ディスプレイ上に配置して動作パターンを作成するオーサリング・システム及び方法に関する。

【０００３】
【従来の技術】電気的若しくは磁気的な作用を用いて人間の動作に似せた運動を行う機械装置のことを「ロボット」という。ロボットの語源は、スラブ語のＲＯＢＯＴＡ(奴隷機械)に由来すると言われている。わが国では、ロボットが普及し始めたのは１９６０年代末からであるが、その多くは、工場における生産作業の自動化・無人化などを目的としたマニピュレータや搬送ロボットなどの産業用ロボット（industrial robot）であった。

【０００４】最近では、イヌやネコのように４足歩行の動物の身体メカニズムやその動作を模したペット型ロボット、あるいは、ヒトやサルなどの２足直立歩行を行う動物の身体メカニズムや動作を模した「人間形」若しくは「人間型」のロボット（humanoid robot）など、脚式移動ロボットやその安定歩行制御に関する研究開発が進展し、実用化への期待も高まってきている。これら脚式移動ロボットは、クローラ式ロボットに比し不安定で姿勢制御や歩行制御が難しくなるが、階段の昇降や障害物の乗り越え等、柔軟な歩行・走行動作を実現できるという点で優れている。

【０００５】アーム式ロボットのように、ある特定の場所に植設して用いるような据置きタイプのロボットは、部品の組立・選別作業など固定的・局所的な作業空間でのみ活動する。これに対し、移動式のロボットは、作業空間は非限定的であり、所定の経路上または無経路上を自在に移動して、所定の若しくは任意の人的作業を代行したり、ヒトやイヌあるいはその他の生命体に置き換わる種々のサービスを提供することができる。

【０００６】脚式移動ロボットの用途の1つとして、産

業活動・生産活動等における各種の難作業の代行が挙げられる。例えば、原子力発電プラントや火力発電プラント、石油化学プラントにおけるメンテナンス作業、製造工場における部品の搬送・組立作業、高層ビルにおける清掃、火災現場その他における救助といったような危険作業・難作業の代行などである。

【０００７】また、脚式移動ロボットの他の用途として、上述の作業支援というよりも、生活密着型、すなわち人間との「共生」あるいは「エンターティンメント」という用途が挙げられる。この種のロボットは、ヒトあるいはイヌ（ペット）などの比較的知性の高い脚式歩行動物の動作メカニズムや四肢を利用した豊かな感情表現をエミュレートする。また、予め入力された動作パターンを単に忠実に実行するだけではなく、相手の言葉や態度（「褒める」とか「叱る」、「叩く」など）に呼応した、生き生きとした応答表現を実現することも要求される。

【０００８】従来の玩具機械は、ユーザ操作と応答動作との関係が固定的であり、玩具の動作をユーザの好みに合わせて変更することはできない。この結果、ユーザは同じ動作しか繰り返さない玩具をやがては飽きてしまうことになる。

【０００９】これに対し、知的なロボットは、動作に起因するモデルを備えており、外部からの音声や画像、触覚などの入力情報に基づいてモデルを変化させて動作を決定することにより、自律的な思考及び動作制御を実現する。ロボットが感情モデルや本能モデルを用意することにより、ロボット自身の感情や本能に従った自律的な行動を表出することができる。また、ロボットが画像入力装置や音声入出力装置を装備し、画像認識処理や音声認識処理を行うことにより、より高度な知的レベルで人間とのリアリスティックなコミュニケーションを実現することも可能となる。

【００１０】また、ユーザ操作などの外部からの刺激を検出したことに応答してこのモデルを変更する、すなわち「学習効果」を付与することによって、ユーザにとって飽きない又は好みに適応した動作パターンを提供することができる。

【００１１】昨今の脚式移動ロボットは高い情報処理能力を備えており、一種の計算機システムとして捉えることができる。したがって、ロボット上で実現される動作パターン、あるいは、複数の基本的な動作パターンの組合せによって構成される高度且つ複雑な一連の動作シーケンスは、コンピュータ・プログラミングと同様の作業によって構築される。

【００１２】また、今後ますますロボットの普及率が高まり、産業界のみならず一般家庭や日常生活にも深く浸透していくことが予想される。とりわけ、エンターティンメント性を追求する製品に関しては、コンピュータやコンピュータ・プログラミングに関する高度な知識を持たない一般消費者層がロボットを購入して使用するケースが多いと予想される。このような一般ユーザにとっても、ロボットの動作シーケンスを対話的な処理により比較的容易且つ効率的に作成・編集するためのことを支援するツール、すなわちオーサリング・システムを提供することが好ましいと考えられる。

【００１３】

【発明が解決しようとする課題】本発明の目的は、ロボットの所定の動作パターンを記述する一連のコマンド／データを作成することができる、優れたオーサリング・システム及び方法を提供することにある。

【００１４】本発明の更なる目的は、ロボットの動作状態を規定する部品の集合を用いて動作パターンを作成することができる、優れたオーサリング・システム及び方法を提供することにある。

【００１５】本発明の更なる目的は、各部品をコンピュータ・ディスプレイ上に配置して動作パターンを作成することができる、優れたオーサリング・システム及び方法を提供することにある。

【００１６】

【課題を解決するための手段】本発明は、上記課題を参酌してなされたものであり、その第１の側面は、ロボットの動作制御プログラムを作成・編集するためのオーサリング・システム又は方法であって、ユーザに対して該ロボットの動作シナリオを作成・編集するための作業場を提供する編集部又は編集ステップと、前記編集部を介して作成・編集されたシナリオを前記ロボット上で解釈可能なプログラム・コードに変換する変換部又は変換ステップと、を具備することを特徴とするオーサリング・システム又は方法である。

【００１７】前記編集部又は編集ステップは、頻繁に使用する動作状態や処理ルーチン又はその雛型を部品化して用意する部品群と、座標指示装置を介した指示操作により前記部品群の各部品を取捨選択して配置する操作画面とをユーザに提供するようにしてもよい。

【００１８】あるいは、前記編集部又は編集ステップは、所定のプログラム言語形式のスクリプトを作成・編集する環境をユーザに提供するようにしてもよい。

【００１９】また、前記変換部又は変換部は、部品の配置により表現された動作シナリオ及び／又はスクリプト形式で記述された動作シナリオを、前記ロボットにおいて解釈可能なニーモニック・コードに変換するようにしてもよい。ここで言うニーモニック・コードの一例はＲＣＯＤＥである。ＲＣＯＤＥの詳細については後述に譲る。

【００２０】また、前記変換部又は変換ステップにおいて変換されたプログラム・コードを前記ロボットに転送するための通信手段又は通信ステップをさらに備えてもよい。通信手段又は通信ステップでは、例えば、ｂｌｕｅｔｏｏｔｈや、１１Ｂなどのような近距離無線データ

通信を適用することができる。

【００２１】ここで言う通信手段又は通信ステップは、前記変換部又は変換ステップにおいて変換されたプログラム･コードを１ステップずつ取り出して前記ロボットに転送するようにしてもよい。これに対し、ロボット側では、インタープリタを用いて受信したプログラム･コードを逐次解釈して実行及び／又はデバッグすることができる。

【００２２】また、通信データのセキュリティを保つため、前記通信手段又は通信ステップは、前記変換部又は変換ステップにおいて変換されたプログラム･コードを暗号化して前記ロボットに転送するようにしてもよい。

【００２３】また、前記通信手段又は通信ステップを介して受信したプログラム･コードを解釈する解釈部又は解釈ステップと、前記解釈部又は解釈ステップによる解釈結果に従って前記ロボットを駆動する駆動制御部又は駆動制御ステップとを前記ロボット上に備えてもよい。

【００２４】このような場合、前記解釈部又は解釈ステップは、受信したプログラム･コードを１ステップ単位で解釈して実行するようにしてもよい。

【００２５】脚式ロボットは一般に、現在の姿勢から直接遷移可能な動作や姿勢と、ある動作や姿勢を経由してなら遷移可能となる姿勢とを持つ。本発明に係るロボットは、現在の姿勢から直接遷移可能な動作や姿勢並びにある動作や姿勢を経由してなら遷移可能となる姿勢などに関する姿勢遷移制限情報をあらかじめ備えていてもよい。

【００２６】このような場合、前記駆動制御部又は駆動制御ステップは、該姿勢遷移制限情報を基に、プログラム･コードの指示内容を姿勢遷移可能な形式に変換することができる。

【００２７】また、かかる姿勢遷移制限情報は、前記ロボットがとり得る姿勢を示すノードと、遷移可能な２つのノード間を結ぶ動作アークで構成される有向グラフ形式で保持することができる。したがって、前記駆動制御部又は駆動制御ステップは、該有向グラフを探索することによって、プログラム･コードの指示内容を姿勢遷移可能な形式に容易に変換することができる。

【００２８】また、本発明の第２の側面は、ロボットの動作制御プログラムを作成･編集するためのオーサリング処理をコンピュータ･システム上で実行するように記述されたコンピュータ･ソフトウェアをコンピュータ可読形式で物理的に格納した記憶媒体であって、前記コンピュータ･ソフトウェアは、ユーザに対して該ロボットの動作シナリオを作成・編集するための作業場を提供する編集ステップと、前記編集ステップを介して作成･編集されたシナリオを前記ロボット上で解釈可能なプログラム･コードに変換する変換ステップと、を具備することを特徴とする記憶媒体である。

【００２９】本発明の第２の側面に係る記憶媒体は、例えば、様々なプログラム･コードを実行可能な汎用コンピュータ･システムに対して、コンピュータ・ソフトウェアをコンピュータ可読な形式で物理的に提供する媒体である。このような媒体は、例えば、ＣＤ（Compact Disc）やＦＤ（Floppy Disc）、ＭＯ（Magneto-Opticaldisc）などの着脱自在で可搬性の記憶媒体である。あるいは、ネットワーク（ネットワークは無線、有線の区別を問わない）などの伝送媒体などを経由してコンピュータ･ソフトウェアを特定のコンピュータ・システムにコンピュータ可読形式で提供することも技術的に可能である。

【００３０】このような記憶媒体は、コンピュータ・システム上で所定のコンピュータ・ソフトウェアの機能を実現するための、コンピュータ・ソフトウェアと記憶媒体との構造上又は機能上の協働的関係を定義したものである。換言すれば、本発明の第２の側面に係る記憶媒体を介して所定のコンピュータ・ソフトウェアをコンピュータ・システムにインストールすることによって、コンピュータ・システム上では協働的作用が発揮され、本発明の第１の側面に係るオーサリング･システム又は方法と同様の作用効果を得ることができる。

【００３１】

【作用】ユーザは、本発明を実現したオーサリング･ツールが提供するＧＵＩ画面を用いてマウス操作によりロボットの規定のシナリオを作成・編集する。あるいは、ユーザは、テキスト・エディタなどを用いて、スクリプト形式でロボットの動作制御プログラムを作成・編集してもよい。

【００３２】オーサリング・ツールは、作成・編集したシナリオやスクリプト形式の動作制御プログラムをＲＣＯＤＥと呼ばれるニーモニック・コードに変換する。ＲＣＯＤＥ動作制御プログラムのデバッグ時には、ＲＣＯＤＥプログラムを１行ごとに取り出して、暗号化して、ｂｌｕｅｔｏｏｔｈや，１１Ｂなどの無線通信手段を利用してロボット側に逐次転送する。

【００３３】他方のロボット側では、ＲＣＯＤＥなどで記述された動作制御プログラムの実行及びデバッグ環境として、インタープリタ／デバッガと、ミドルウェアと、ドライバなどを備えている。

【００３４】インタープリタは、ＲＣＯＤＥ形式で記述されたプログラムを１行ずつ読み込んで解釈して実行する高水準言語プログラムである。デバッガは、ＲＣＯＤＥプログラム中の誤り（バグ）を発見して、修正する作業を支援するプログラムである。ミドルウェアは、ロボットにおける歩行その他の脚式動作などの制御や、カメラからの入力画像の認識処理や、マイクからの音声入力の認識処理などを行う処理エンジンである。

【００３５】ミドルウェアは、ロボット１が遷移可能な姿勢及び遷移する際の動作を、例えば有向グラフの形式であらかじめ登録しており、有向グラフに従って上位プ

ログラムからの行動指令を姿勢遷移情報に変換して、ドライバに対する具体的な動作指示を発行するようになっている。

【００３６】したがって、一般ユーザは、オーサリング・ツールを用いてＧＵＩ画面上でロボット１の動作シナリオを編集したり、あるいはスクリプト形式で動作制御プログラムを記述する限りにおいて、現実のロボットにおける姿勢・動作遷移特性のような詳細なハードウェア情報を気にしなくて済む。また、ミドルウェア・レベルにおいてＲＣＯＤＥコマンド・レベルでの論理的な姿勢・動作指示とハードウェア上の姿勢・動作遷移特性との差異を吸収するので、ロボットの姿勢・動作遷移特性に反していることはＲＣＯＤＥプログラムのデバッグの対象とはならない。

【００３７】本発明のさらに他の目的、特徴や利点は、後述する本発明の実施例や添付する図面に基づくより詳細な説明によって明らかになるであろう。

【００３８】

【発明の実施の形態】以下、図面を参照しながら本発明の実施例を詳解する。

【００３９】Ａ．オーサリング・システムの構成

図１には、本発明を実施に供される、四肢による脚式歩行を行う移動ロボット１の外観構成を示している。図示の通り、該ロボット１は、四肢を有する動物の形状や構造をモデルにして構成された多関節型の移動ロボットである。とりわけ本実施例の移動ロボット１は、愛玩動物の代表例であるイヌの形状及び構造を模してデザインされたペット型ロボットという側面を有し、例えば人間の住環境において人間と共存するとともに、ユーザ操作に応答した動作表現することができる。

【００４０】移動ロボット１は、胴体部ユニット２と、頭部ユニット３と、尻尾４と、四肢すなわち脚部ユニット６Ａ～６Ｄで構成される。

【００４１】頭部ユニット３は、ロール、ピッチ及びヨーの各軸方向（図示）の自由度を持つ首関節７を介して、胴体部ユニット２の略前上端に配設されている。また、頭部ユニット３には、イヌの「目」に相当するＣＣＤ（Charge Coupled Device：電荷結合素子）カメラ１５と、「耳」に相当するマイクロフォン１６と、「口」に相当するスピーカ１７と、触感に相当するタッチセンサ１８が搭載されている。これら以外にも、生体の五感を構成するセンサを含んでいても構わない。

【００４２】尻尾４は、ロール及びピッチ軸の自由度を持つ尻尾関節８を介して、胴体部ユニット２の略後上端に湾曲若しくは揺動自在に取り付けられている。

【００４３】脚部ユニット６Ａ及び６Ｂは前足を構成し、脚部ユニット６Ｃ及び６Ｄは後足を構成する。各脚部ユニット６Ａ～６Ｄは、それぞれ、大腿部ユニット９Ａ～９Ｄ及び脛部ユニット１０Ａ～１０Ｄの組み合わせで構成され、胴体部ユニット２底面の前後左右の各隅部に取り付けられている。大腿部ユニット９Ａ～９Ｄは、ロール、ピッチ、ヨーの各軸の自由度を持つ股関節１１Ａ～１１Ｄによって、胴体部ユニット２の各々の所定部位に連結されている。また、大腿部ユニット９Ａ～９Ｄと脛部ユニット１０Ａ～１０Ｄの間は、ロール及びピッチ軸の自由度を持つ膝関節１２Ａ～１２Ｄによって連結されている。

【００４４】図示のように構成された移動ロボット１は、後述する制御部からの指令により各関節アクチュエータを駆動することによって、例えば、頭部ユニット３を上下左右に振らせたり、尻尾４を振らせたり、各足部ユニット６Ａ～６Ｄを協調的に駆動させて歩行や走行などの動作を実現することができる。

【００４５】なお、移動ロボット１の関節自由度は、実際には各軸毎に配備され関節アクチュエータ（図示しない）の回転駆動によって提供される。また、移動ロボット１が持つ関節自由度の個数は任意であり、本発明の要旨を限定するものではない。

【００４６】図２には、移動ロボット１の電気・制御系統の構成図を模式的に示している。同図に示すように、移動ロボット１は、全体の動作の統括的制御やその他のデータ処理を行う制御部２０と、入出力部４０と、駆動部５０と、電源部６０とで構成される。以下、各部について説明する。

【００４７】入出力部４０は、入力部として移動ロボット１の目に相当するＣＣＤカメラ１５や、耳に相当するマイクロフォン１６、触感に相当するタッチセンサ１８など、五感に相当する各種のセンサを含む。また、出力部として、口に相当するスピーカ１７などを装備している。これら出力部は、脚などによる機械運動パターン以外の形式でロボット１からのユーザ・フィードバックを表現することができる。

【００４８】移動ロボット１は、カメラ１５を含むことで、作業空間上に存在する任意の物体の形状や色彩を認識することができる。また、移動ロボット１は、カメラのような視覚手段の他に、赤外線、音波、超音波、電波などの発信波を受信する受信装置をさらに備えていてもよい。この場合、各伝送波を検知するセンサ出力に基づいて発信源からの位置や向きを計測することができる。

【００４９】駆動部５０は、制御部２０が指令する所定の運動パターンに従って移動ロボット１の機械運動を実現する機能ブロックであり、首関節７、尻尾関節８、股関節１１Ａ～１１Ｄ、膝関節１２Ａ～１２Ｄなどのそれぞれの関節におけるロール、ピッチ、ヨーなど各軸毎に設けられた駆動ユニットで構成される。図示の例では、移動ロボット１はｎ個の関節自由度を有し、したがって駆動部５０はｎ個の駆動ユニットで構成される。各駆動ユニットは、所定軸回りの回転動作を行うモータ５１と、モータ５１の回転位置を検出するエンコーダ５２と、エンコーダ５２の出力に基づいてモータ５１の回転

位置や回転速度を適応的に制御するドライバ５３の組み合わせで構成される。

【００５０】電源部６０は、その字義通り、移動ロボット１内の各電気回路等に対して給電を行う機能モジュールである。本実施例に係る移動ロボット１は、バッテリを用いた自律駆動式であり、電源部６０は、充電バッテリ６１と、充電バッテリ６１の充放電状態を管理する充放電制御部６２とで構成される。

【００５１】充電バッテリ６１は、例えば、複数本のニッケル・カドミウム電池セルをカートリッジ式にパッケージ化した「バッテリ・パック」の形態で構成される。

【００５２】また、充放電制御部６２は、バッテリ６１の端子電圧や充電／放電電流量、バッテリ６１の周囲温度などを測定することでバッテリ６１の残存容量を把握し、充電の開始時期や終了時期などを決定するようになっている。充放電制御部６２が決定する充電の開始及び終了時期は制御部２０に通知され、移動ロボット１が充電オペレーションを開始及び終了するためのトリガとなる。充電オペレーションの詳細については後述に譲る。

【００５３】制御部２０は、「頭脳」に相当し、例えば移動ロボット１の頭部ユニット３あるいは胴体部ユニット２に搭載される。

【００５４】図３には、制御部２０の構成をさらに詳細に図解している。同図に示すように、制御部２０は、メイン・コントローラとしてのＣＰＵ（Central Processing Unit）２１が、メモリその他の各回路コンポーネントや周辺機器とバス接続された構成となっている。バス２７上の各装置にはそれぞれに固有のアドレス（メモリ・アドレス又はＩ／Ｏアドレス）が割り当てられており、ＣＰＵ２１はアドレス指定することでバス２８上の特定の装置と通信することができる。

【００５５】ＲＡＭ（Random Access Memory）２２は、ＤＲＡＭ（Dynamic RAM）などの揮発性メモリで構成された書き込み可能メモリであり、ＣＰＵ２１が実行するプログラム・コードをロードしたり、作業データの一時的な保存のために使用される。

【００５６】ＲＯＭ（Read Only Memory）２３は、プログラムやデータを恒久的に格納する読み出し専用メモリである。ＲＯＭ２３に格納されるプログラム・コードには、移動ロボット１の電源投入時に実行する自己診断テスト・プログラムや、移動ロボット１の動作を規定する制御プログラムなどが挙げられる。

【００５７】ロボット１の制御プログラムには、カメラ１５やマイクロフォン１６などのセンサ入力を処理する「センサ入力処理プログラム」、センサ入力と所定の動作モデルとに基づいて移動ロボット１の行動すなわち運動パターンを生成する「行動命令プログラム」、生成された運動パターンに従って各モータの駆動やスピーカ１７の音声出力などを制御する「駆動制御プログラム」などが含まれる。生成される運動パターンには、通常の歩行運動や走行運動以外に、「お手」、「お預け」、「お座り」や、「ワンワン」などの動物の鳴き声の発声などエンターティンメント性の高い動作を含んでいてもよい。

【００５８】また、ロボット１のその他の制御プログラムとして、オーサリング・ツールを用いて作成・編集された各種の動作シーケンス・プログラムが含まれる。オーサリング・ツールは、例えば外部のコンピュータ・システム上で所定のソフトウェア実行環境下で起動するが、オーサリング・ツール並びに該ツール上で作成・編集されるプログラムついては後に詳解する。

【００５９】不揮発性メモリ２４は、例えばＥＥＰＲＯＭ（Electrically Erasable and Programmable ROM）のように、電気的に消去再書き込みが可能なメモリ素子で構成され、逐次更新すべきデータを不揮発的に保持するために使用される。逐次更新すべきデータには、例えば、移動ロボット１の行動パターンを規定するモデルなどが挙げられる。

【００６０】インターフェース２５は、制御部２０外の機器と相互接続し、データ交換を可能にするための装置である。インターフェース２５は、例えば、カメラ１５やマイクロフォン１６、スピーカ１７との間でデータ入出力を行う。また、インターフェース２５は、駆動部５０内の各ドライバ５３－１…との間でデータやコマンドの入出力を行う。また、インターフェース２５は、電源部６０との間で充電開始及び充電終了信号の授受を行うこともできる。

【００６１】インターフェース２５は、ＲＳ（Recommended Standard）－２３２Ｃなどのシリアル・インターフェース、ＩＥＥＥ（Institute of Electrical and electronics Engineers）１２８４などのパラレル・インターフェース、ＵＳＢ（Universal Serial Bus）インターフェース、ｉ－Ｌｉｎｋ（ＩＥＥＥ１３９４）インターフェース、ＳＣＳＩ（Small Computer System Interface）インターフェースなどのような、コンピュータの周辺機器接続用の汎用インターフェースを備え、ローカル接続された外部機器との間でプログラムやデータの移動を行うようにしてもよい。

【００６２】また、インターフェース２５の１つとして赤外線通信（ＩｒＤＡ）インターフェースを備え、外部機器と無線通信を行うようにしてもよい。赤外線通信のための送受信部は、例えば頭部ユニット２や尻尾３など、移動ロボット１本体の先端部に設置されることが受信感度の観点から好ましい。

【００６３】さらに、制御部２０は、無線通信インターフェース２６ネットワーク・インターフェース・カード（ＮＩＣ）２７を含み、"ｂｌｕｅｔｏｏｔｈ"や"．１１Ｂ"のような近接無線通信、あるいはＬＡＮ（Local Area Network：例えばＥｔｈｅｒｎｅｔ（登録商標））やインターネットを経由して、外部のホスト・コンピュ

ータ１００とデータ通信を行うことができる。

【００６４】このようなロボットとホストコンピュータ間のデータ通信の目的は、遠隔のコンピュータ資源を用いて移動ロボット１の動作をリモート・コントロールすることである。また、該データ通信の他の目的は、動作モデルやその他のプログラム・コードなどロボット１の動作制御に必要なデータやプログラムをネットワーク経由で移動ロボット１に供給することにある。また、該データ通信の他の目的は、ホスト・コンピュータ１００上でオーサリング・ツールを用いて作成・編集したロボット動作制御用プログラムのダウンロードや、このような動作制御用プログラムのホスト・コンピュータ１００とロボット１との協働的動作によるリアルタイムのデバッグ処理である。但し、オーサリング・ツールやデバッグ処理については後に後述する。

【００６５】また、制御部２０は、テンキー及び／又はアルファベット・キーからなるキーボード２９を備えておいてもよい。キーボード２９は、作業現場での直接的なコマンド入力のために使用する他、パスワードなどの所有者認証情報の入力に用いられる。

【００６６】本実施例に係るロボット１は、制御部２０が所定の制御プログラムを実行することによって自律的な動作を行うことができる。また、画像入力（すなわちカメラ１５）、音声入力（すなわちマイク１６）、タッチセンサ１８などの人間や動物の五感に相当する入力装置を備えるとともに、これら外部入力に応答した理性的又は感情的な動作を実行するインテリジェンスを備えている。

【００６７】図１～図３に示すように構成された移動ロボット１は、以下のような特徴がある。すなわち、

【００６８】（１）ある姿勢から他の姿勢へ遷移するように指示されたとき、各姿勢間を直接に遷移せず、あらかじめ用意された無理のない中間的な姿勢を経由して遷移する。

（２）姿勢遷移で任意の姿勢に到達したときに通知を受け取ることができる。

（３）頭部、足部、尻尾部などの各ユニット単位で姿勢を独立して管理し姿勢制御することができる。すなわち、ロボット１の全体の姿勢とは別に各ユニット毎に姿勢を管理することができる。

（４）動作命令の動作の詳細を示すためのパラメータを渡すことができる。

【００６９】図３に示すように、本実施例に係る移動ロボット１は、ネットワーク経由で外部のホスト・コンピュータ１００と相互接続されている。あるいは、ホスト・コンピュータ１００とは、無線通信（例えば、ｂｌｕｅｔｏｏｔｈや．１１Ｂ近距離無線データ通信）やその他の通信手段によって接続されていてもよい。

【００７０】ホスト・コンピュータ１００上では、所定のソフトウェア実行環境が構築され、該環境下では、オーサリング・ツールを起動して、対話的な処理によりロボット１の動作シーケンスを比較的容易且つ効率的に作成することができる。オーサリング・ツールの詳細については後述する。

【００７１】図４には、ホスト・コンピュータ１００のハードウェア構成例を模式的に図解している。以下、コンピュータ１００内の各部について説明する。

【００７２】システム１００のメイン・コントローラであるＣＰＵ（Central Processing Unit）１０１は、オペレーティング・システムＯＳの制御下で、各種のアプリケーションを実行するようになっている。ＯＳは、より好ましくはＧＵＩ（Graphical User Interface）環境を提供するが、例えば、ＵＮＩＸ（登録商標）、又は、米Ｍｉｃｒｏｓｏｆｔ社のＷｉｎｄｏｗｓ９８／ＮＴでよい。

【００７３】図示の通り、ＣＰＵ１０１は、バス１０７によって他の機器類（後述）と相互接続されている。バス１０７上の各機器にはそれぞれ固有のメモリ・アドレス又はＩ／Ｏアドレスが付与されており、ＣＰＵ１０１はこれらアドレスによって機器アクセスが可能となっている。バス１０７の一例はＰＣＩ（Peripheral Component Interconnect）バスである。

【００７４】メモリ１０２は、プロセッサ１０１において実行されるプログラム・コードを格納したり、実行中の作業データを一時保管するために使用される記憶装置である。同図に示すメモリ１０２は、不揮発及び揮発メモリ双方を含むものと理解されたい。

【００７５】ディスプレイ・コントローラ１０３は、プロセッサ１０１が発行する描画命令を実際に処理するための専用コントローラであり、例えばＳＶＧＡ（Super Video Graphic Array）又はＸＧＡ（eXtended Graphic Array）相当のビットマップ描画機能をサポートする。ディスプレイ・コントローラ１０３において処理された描画データは、例えばフレーム・バッファ（図示しない）に一旦書き込まれた後、表示装置１１１に画面出力される。表示装置１１１は、例えば、ＣＲＴ（Cathode Ray Tube）ディスプレイや、液晶表示ディスプレイ（Liquid Crystal Display）などである。

【００７６】入力機器インターフェース１０４は、キーボード１１２やマウス１１３などのユーザ入力機器をシステム１００に接続するための装置である。入力機器インターフェース１０４は、キーボード１１２によるキー入力又はマウス１１３を介した座標指示入力に応答して、プロセッサ１０１に対して割り込みを発生する。

【００７７】ネットワーク・インターフェース１０５は、Ｅｔｈｅｒｎｅｔなどの所定の通信プロトコルに従って、システム１００をＬＡＮ（Local Area Network）などのネットワークに接続したり、あるいはｂｌｕｅｔｏｏｔｈや．１１Ｂのような近距離無線データ通信に接続することができる。ネットワーク・インターフェース

１０５は、一般に、ＬＡＮアダプタ・カードの形態で提供され、マザーボード（図示しない）上のＰＣＩバス・スロットの装着して用いられる。

【００７８】図３に示す例では、ホスト・コンピュータ１００は、無線データ通信やネットワーク経由でロボット１と相互接続されているが、勿論、他の通信手段によって両者が接続されていてもよい。

【００７９】またネットワーク上では、複数のホスト・コンピュータ（図示しない）がトランスペアレントな状態で接続され、分散コンピューティング環境が構築されている。ネットワーク上では、ソフトウェア・プログラムやデータ・コンテンツなどの配信が行われる。例えば、本実施例に係るオーサリング・ツールや、該オーサリング・ツールで作成・編集したロボット用動作シーケンス・プログラムなどを、ネットワーク経由で配信することができる。また、このようなプログラム／データの配信サービスを有料又は無料で行ってもよい。

【００８０】外部機器インターフェース１０６は、ハード・ディスク・ドライブ（ＨＤＤ）１１４やメディア・ドライブ１１５などの外部装置をシステム１００に接続するための装置である。外部機器インターフェース１０６は、例えば、ＩＤＥ（Integrated Drive Electronics）やＳＣＳＩ（Small Computer System Interface）などのインターフェース規格に準拠する。

【００８１】ＨＤＤ１１４は、記憶担体としての磁気ディスクを固定的に搭載した外部記憶装置であり（周知）、記憶容量やデータ転送速度などの点で他の外部記憶装置よりも優れている。ソフトウェア・プログラムを実行可能な状態でＨＤＤ１１６上に置くことをプログラムのシステムへの「インストール」と呼ぶ。通常、ＨＤＤ１１４には、プロセッサ５１１が実行すべきオペレーティング・システムのプログラム・コードや、アプリケーション・プログラム、デバイス・ドライバなどが不揮発的に格納されている。例えば、本実施例に係るオーサリング・ツールや、該オーサリング・ツールを用いて作成・編集したロボット用動作シーケンス・プログラムを、ＨＤＤ１１４上にインストールすることができる。

【００８２】また、メディア・ドライブ１１５は、ＣＤ（Compact Disc）やＭＯ（Magneto-Optical disc）、ＤＶＤ（Digital Versatile Disc）などの可搬型メディアを装填して、データ記録面にアクセスするための装置である。可搬型メディアは、主として、ソフトウェア・プログラムやデータ・ファイルなどをコンピュータ可読形式のデータとしてバックアップすることや、これらをシステム間で移動（販売・流通・配布を含む）する目的で使用される。例えば、本実施例に係るオーサリング・ツールや、該オーサリング・ツールを用いて作成したロボット用動作シーケンス・プログラムなどを、これら可搬型メディアを利用して機器間で物理的に流通・配布することができる。

【００８３】なお、図４に示すようなホスト・コンピュータ１００の一例は、米ＩＢＭ社のパーソナル・コンピュータＰＣ／ＡＴ（Personal Computer/Advanced Technology）の互換機又は後継機である。勿論、他のアーキテクチャを備えた計算機システムを本実施例に係るホスト・コンピュータ１００に適用することも可能である。

【００８４】本実施例では、ロボット１の所定の動作パターンを記述する一連のコマンド／データからなる動作制御プログラムを、ホスト・コンピュータ１００上で起動したオーサリング・ツールを用いて作成・編集する。また、このオーサリング・ツールを用いて作成・編集した動作制御プログラムを、例えばｂｌｕｅｔｏｏｔｈや．１１Ｂなどの無線通信手段を用いてロボット１側に転送して、ホスト・コンピュータ１００とロボット１との協働的動作によりデバッグ処理を行うようになっている。すなわち、ホスト・コンピュータ１００とロボット１の誘起的な結合により、ロボット１の動作制御プログラムのためのオーサリング・システムが構築される。

【００８５】図５には、オーサリング・システムの全体構成を模式的に図解している。

【００８６】ホスト・コンピュータ１００側では、ユーザは、オーサリング・ツールが提供するＧＵＩ（Graphical User Interface）画面を用いてマウス操作によりロボット１の規定のシナリオを作成・編集することができる。シナリオ作成用のＧＵＩ画面並びに該画面上の操作については後述する。あるいは、ユーザは、テキスト・エディタなどを用いて、スクリプト形式（例えばＣ言語などの高級言語形式）でロボット１の動作制御プログラムを作成・編集することができる。

【００８７】オーサリング・ツールは、ユーザがＧＵＩ画面上で作成・編集したシナリオや、テキスト・エディタ上で作成・編集したスクリプト形式の動作制御プログラムを、"ＲＣＯＤＥ"と呼ばれるアセンブラのようなニーモニック・コードに変換する。

【００８８】ＲＣＯＤＥとは、簡単なコマンドでロボット１を制御するために策定されたプログラム言語であり、ＩＦやＧＯなどの基本的な制御構造も備えているので、ロボット制御用最低水準スクリプト言語としても使用することができる。但し、ＲＣＯＤＥの詳細については後に詳解する。

【００８９】ホスト・コンピュータ１００上で作成・編集されたＲＣＯＤＥ動作制御プログラムは、例えば、メモリ・スティックなどのメディアを利用してロボット１側に移動することができる。また、ＲＣＯＤＥ動作制御プログラムのデバッグ時には、ＲＣＯＤＥプログラムを１行ごとに取り出して、暗号化して、ｂｌｕｅｔｏｏｔｈや．１１Ｂなどの無線通信手段を利用してロボット１側に逐次転送するようになっている。

【００９０】他方、ロボット１側では、ＲＣＯＤＥなどで記述された動作制御プログラムの実行及びデバッグ環

境として、インタープリタ／デバッガと、ミドルウェアと、ドライバと、オペレーティング・システム（ＯＳ）とを備えている。

【００９１】インタープリタは、ＲＣＯＤＥ形式で記述されたプログラムを１行ずつ読み込んで解釈して実行する高水準言語プログラムである。但し、デバッグ時などにおいて、ホスト・コンピュータ１００側から暗号化された形式でＲＣＯＤＥプログラムが送信される場合には、インタープリタは、これを一旦復号化してから解釈・実行を行う。

【００９２】デバッガは、ＲＣＯＤＥプログラム中の誤り（バグ）を発見して、修正する作業を支援するプログラムである。すなわち、プログラムを指定した行で実行を止めたり、そのときのメモリや変数の内容を参照することができる。

【００９３】ミドルウェアは、ロボット１における歩行その他の脚式動作などの制御や、カメラ１５からの入力画像の認識処理や、マイク１６からの音声入力の認識処理などを行う処理エンジンである。

【００９４】ドライバは、各関節アクチュエータやその他のハードウェアの操作を行うためのプログラム・コードである。

【００９５】本実施例では、ミドルウェア並びにドライバは、オブジェクト指向プログラムによって実装されている。オブジェクト指向に基づくソフトウェアは、基本的に、データとそのデータに対する処理手続きとを一体化させた「オブジェクト」というモジュール単位で扱われる。また、必要に応じて複数のオブジェクトを作成したり組み合わせることで１つのソフトウェアが完成する。一般に、オブジェクト指向によれば、ソフトウェアの開発と保守が効率化されると考えられている。

【００９６】オペレーティング・システム（ＯＳ）は、これらオブジェクト間のデータ通信の管理や、その他のプログラム実行に関する制御を行う。ＯＳもオブジェクト指向プログラムにより実装される。

【００９７】Ｂ．オーサリング・ツールを用いたロボット用動作プログラムの作成・編集

次いで、本実施例に係るオーサリング・ツールが提供する、ユーザがロボット１の動作シナリオを作成・編集するためのＧＵＩ画面について説明する。図６には、該ＧＵＩ画面の構成を図解している。

【００９８】同図に示すように、該ＧＵＩ画面は、マウスによるクリック、ドラッグ＆ドロップ操作を基調として編集作業を行う編集ウィンドウと、編集作業において頻繁に使用される動作手順を部品化して提供する部品ウィンドウを含んでいる。

【００９９】編集ウィンドウは、最上段より順に、タイトル・バー、メニュー・バー、ツール・バー、編集領域が配設されている。

【０１００】メニュー・バーは、ユーザがクリック操作により選択可能な処理の一覧を横方向に配列した領域であり、この例では、「ファイル（Ｆ）」、「編集（Ｅ）」、「表示（Ｖ）」、「挿入（Ｉ）」、「グループ（Ｇ）」という各メニュー項目が用意されている。あるメニュー項目を選択すると、さらに該当するプルダウン・メニューが出現する。

【０１０１】図７には、メニュー項目「ファイル」に関するプルダウン・メニューを図解している。同図に示すように、該プルダウン・メニュー中には、「新規作成」、「開く」、「上書き保存」、「名前を付けて保存」、「メモリスティックイメージの作成」、「メモリスティックイメージの転送」、「終了」という各サブメニューが配列されている。

【０１０２】「新規作成」を選択すると、ロボットの一連の動作を記述するシナリオ（以下、「プロジェクト」とも言う）を新規に作成する。既に編集中のプロジェクトが編集ウィンドウ上に存在する場合には、その保存を促すダイアログが出現する。図６に示す編集ウィンドウは新規プロジェクトを作成したときの初期画面であり、プロジェクトの先頭と最後尾にそれぞれ該当する"ＳＴＡＲＴ"ボックス及び"ＥＮＤ"ボックスのみからなるメイン・グループが編集領域に表示されている。

【０１０３】サブメニュー「開く」を選択すると、ファイルとして保存されたプロジェクトを編集ウィンドウ上に開く。既に編集中のプロジェクトが編集ウィンドウ上に存在する場合には、その保存を促すダイアログが表示される。本実施例では、プロジェクトは、拡張子"ａｐｋ"を持つファイルとして扱われるものとする。

【０１０４】サブメニュー「上書き保存」を選択すると、同名のファイルに編集中のプロジェクトを保存する。このとき、ファイル名を尋ねるダイアログは出現しないものとする。

【０１０５】サブメニュー「名前を付けて保存」を選択すると、ファイル名を尋ねるダイアログが表示される。該ダイアログ上で、ユーザは保存先のディレクトリ名とファイル名を指定することができ、指定されたディレクトリにそのファイル名でプロジェクトが保存される。

【０１０６】サブメニュー「メモリスティックイメージの作成」を選択すると、メモリ・スティック（又はこれと等価な記憶メディア）に書き込むためのイメージ・ファイルを作成する。本実施例では、イメージ・ファイルは、元のａｐｋ形式のファイルと同じディレクトリに"ａｉｍ"という拡張子で保存されるものとする。

【０１０７】サブメニュー「メモリスティックイメージの転送」を選択すると、メモリ・スティック上のイメージ・ファイルを指定されたフルパスに生成する。

【０１０８】サブメニュー「終了」を選択すると、実行中のオーサリング・ツールを終了する。

【０１０９】また、図８には、メニュー項目「グループ」に関するプルダウン・メニューを図解している。同

図に示すように、該プルダウン・メニュー中には、「新規グループ作成」、「このグループの削除」、「グループ一覧」、「グループ階層構造」、「全てのグループを開く」、「全てのグループを閉じる」という各サブメニューが配列されている。

【０１１０】サブメニュー「新規グループ作成」を選択すると、新規グループの作成を行う。

【０１１１】サブメニュー「このグループの削除」を選択すると、現在表示されているグループの削除を行う。但し、グループを削除しても、そのグループを呼び出しているグループ・ボックスは削除されない。

【０１１２】サブメニュー「グループ一覧」を選択すると、グループの一覧を表示する。表示されたグループの名前をダブルクリックすることで、そのグループを開くことができる。

【０１１３】サブメニュー「グループ階層構造」を選択すると、グループの階層構造（呼び出し構造）を表示する。表示されたグループの名前をダブルクリックすることで、そのグループを開くことができる。

【０１１４】サブメニュー「全てのグループを開く」を選択すると、すべてのグループのウィンドウを開く。

【０１１５】サブメニュー「全てのグループを閉じる」を選択すると、すべてのグループのウィンドウを閉じる（但し、メイン・グループは除く）。

【０１１６】また、図６に示す例では、ツール・バー内には、「通常モード」、「接続モード」、「動作ボックス」、「分岐ボックス」、「グループ・ボックス」という各ツール・ボタンが配設されている。

【０１１７】ボタン「通常モード」を選択すると、マウス操作が通常モードに遷移する。例えば、ボックスに対するドラッグ動作は、ボックスの移動として機能する。

【０１１８】ボタン「接続モード」を選択すると、マウス操作が接続モードに遷移する。すなわち、ボックスに対するドラッグ動作は、ボックス間の接続として機能する。接続元から接続先のボックスに対してドラッグ動作を行えばよい。また、分岐ボックスからの接続である場合、分岐条件を指定するダイアログが表示される。

【０１１９】ボタン「動作ボックス」を選択すると、動作（Ａｃｔｉｏｎ）ボックスが下方の編集領域に挿入される。

【０１２０】ボタン「分岐ボックス」を選択すると、分岐（Ｂｒａｎｃｈ）ボックスが下方の編集領域に挿入される。

【０１２１】ボタン「グループ・ボックス」を選択すると、グループ・ボックスが下方の編集領域に挿入される。

【０１２２】ユーザは、図６に示す編集領域上で、クリック、ドラッグなどのマウス操作や、マウス操作とキー入力の組合せにより、処理を指定することができる。

【０１２３】例えば、マウスの左ボタンのクリックによりボックスを選択することができる。

【０１２４】また、マウスの左ボタンをダブル・クリックしたときは、プロパティを表示し、及び／又は、グループを開く。

【０１２５】また、通常モード下でマウスの左ボタンを押下してドラッグしたとき、マウス・カーソルがボックス上にあったときには、ボックスの移動を行い、マウス・カーソルがボックス外にあったときには範囲選択を行う。また、接続モード下でマウスの左ボタンを押下してドラッグしたときには、マウス・カーソルがボックス上にあるときにはボックスどうしの接続を行う。

【０１２６】また、通常モード下で［Ｃｔｒｌ］キー押下とマウスの左ボタンの押下／ドラッグの組合せを行うと、マウス・カーソルがボックス上にあったときにはボックスの接続を行い、マウス・カーソルがボックス外にあったときには範囲指定を行う。また、接続モード下で［Ｃｔｒｌ］キー押下とマウスの左ボタンの押下／ドラッグの組合せを行うと、マウス・カーソルがボックス上にあるときにはボックスの移動を行う。

【０１２７】また、［Ｓｈｉｆｔ］キー押下とマウスの左ボタン押下／ドラッグの組合せを行うと、編集ウィンドウのリサイズを行う。

【０１２８】また、マウスの右ボタンをクリックすると、コンテキスト・メニューが表示される。

【０１２９】なお、マウスのドラッグ操作は、［Ｅｓｃ］キーの押下でキャンセルされる。

【０１３０】次いで、図６に示すような、オーサリング・ツールが提供するＧＵＩ画面上における基本操作手順について説明する。

【０１３１】まず、メニュー［ファイル］のサブメニュー［新規作成］を選択することにより、新規プロジェクトを作成する。新規プロジェクトの画面は、図６に示すように、プロジェクトの先頭と最後尾にそれぞれ該当する"ＳＴＡＲＴ"ボックス及び"ＥＮＤ"ボックスのみからなるメイン・グループが編集領域に表示されている。

【０１３２】次いで、メニュー［挿入］又はツール・バー内のボタン［Ａ］、［Ｂ］、［Ｇ］を選択することにより、編集ウィンドウ上に該当するボックスを配置することができる。一旦編集ウィンドウ上に配置されたボックスは、マウス操作により設置位置を移動することができる。

【０１３３】図９には、編集ウィンドウ上に動作ボックス［Ａ］及び分岐ボックス［Ｂ］が各１個配置されている様子を図解している。各ボックスには、デフォルトの名前"Ｂｏｘ００１"並びに"Ｂｏｘ００２"が付されているが、これはユーザが後で変更することができる。

【０１３４】次いで、メニュー［編集］のサブメニュー［マウス：接続モード］を選択するか、又は、ツール・バー内の［接続モード］ボタンを選択することにより、マウス操作を接続モードに切り換える。そして、マウス

操作により、シナリオ進行上の前後関係にあるボックスどうしを接続する。

【０１３５】図１０には、図９に示す編集ウィンドウ上で、[ＳＴＡＲＴ]ボックスと動作ボックス［Ａ］、並びに、動作ボックス［Ａ］と分岐ボックス［Ｂ］を接続した様子を図解している。

【０１３６】ボックスどうしを接続する際、分岐ボックス［Ｂ］からの接続に関しては、分岐上辺を指定するためのダイアログ（図示しない）が自動的に開くようになっているので、ユーザは、ダイアログ内の指示に従って条件を指定していけばよい。但し、分岐ボックスの条件指定は、次のボックスへの接続時よりも後回しにはできないものとし、また、条件を完全に指定しないと接続が行われない。

【０１３７】また、ユーザは、あるボックス上でマウスの左ボタンをダブルクリックすることで、当該ボックスの処理動作の詳細を指定するためのダイアログが開くようになっている。また、グループ･ボックスをダブルクリックした場合には、そのグループが別の編集ウィンドウとして開くようになっている（後述）。

【０１３８】本実施例では、各ボックスの処理動作は、ＲＣＯＤＥと呼ばれる、アセンブラのようなニーモニック・コードによって記述される。ＲＣＯＤＥとは、簡単なコマンドでロボット１を制御するために策定されたプログラム言語であるが、ＲＣＯＤＥの詳細については後に詳解する。

【０１３９】図１１には、動作ボックスの詳細を指定するためのダイアログを図解している。このダイアログは、例えば図９又は図１０に示すような編集ウィンドウ上で、所望の動作ボックス［Ａ］をダブルクリックすることによって呼び出すことができる。

【０１４０】ユーザは、図１１に示すようなダイアログ上で、名前フィールドに文字列を入力することによって動作ボックスの名前を指定することができる。また、コメント･フィールド上に、該動作ボックスに関するコメントを記入することができる。

【０１４１】さらに、Ａｃｔｉｏｎ、Ｐａｒｔ、Ｓｏｕｎｄ、Ｖｏｌｕｍｅ＃などのコンボボックス内に、ＲＣＯＤＥのコマンド名やコマンド･パラメータを直接入力するか、又は、該ボックスの右端の▼ボタンを押して出現するリスト･ボックス（図示しない）から所望のコマンド又はコマンド･パラメータを選択することによって、１行すなわち１ステップ分のＲＣＯＤＥコマンドを設定することができる。

【０１４２】これらＡｃｔｉｏｎ、Ｐａｒｔ、Ｓｏｕｎｄ、Ｖｏｌｕｍｅ＃などのコンボボックスを利用して１ステップ分のＲＣＯＤＥコマンドを設定し、さらに［追加（Ａ）］ボタンをクリックすると、コマンド･リスト上に順次登録される。

【０１４３】また、このコマンド･リスト上で所定の行を選択してから、［変更（Ｍ）］ボタンをクリックすることによって、当該行は変更対象となり、その設定内容がＡｃｔｉｏｎ、Ｐａｒｔ、Ｓｏｕｎｄ、Ｖｏｌｕｍｅ＃などの各コンボボックスに表示される。また、このコマンド･リスト上で所定の行を選択してから、［削除（Ｄ）］ボタンをクリックすることによって、当該行をコマンド･リストから削除することができる。

【０１４４】そして、このダイアログ上で動作ボックスの詳細の指定を完了すると、［閉じる（Ｃ）］ボタンをクリックすることによって、画面を介した指定内容がＲＣＯＤＥで記述された処理ルーチンとして登録されるとともに、ダイアログは閉じる。

【０１４５】また、図１２には、分岐ボックスの詳細を指定するためのダイアログを図解している。このダイアログは、例えば図９又は図１０に示すような編集ウィンドウ上で、所望の分岐ボックス［Ｂ］をダブルクリックすることによって呼び出すことができる。

【０１４６】ユーザは、図１２に示すようなダイアログ上で、名前フィールドに文字列を入力することによって分岐ボックスの名前を指定することができる。図示の例では、当該分岐ボックスの名前として「モード分岐」が記入されている。また、コメント･フィールド上に、該分岐ボックスに関するコメントを記入することができる。

【０１４７】さらに、Ｔｙｐｅ、Ｖａｒｉａｂｌｅなどのコンボボックス内に、ＲＣＯＤＥのコマンド名やコマンド･パラメータを直接入力するか、又は、該ボックスの右端の▼ボタンを押して出現するリスト･ボックス（図示しない）から所望のコマンド又はコマンド･パラメータを選択することによって、当該分岐ボックスの条件判断を記述するＲＣＯＤＥコマンドを設定することができる。

【０１４８】これらＴｙｐｅ、Ｖａｒｉａｂｌｅなどのコンボボックスを利用して分岐条件ののＲＣＯＤＥコマンドを設定し、さらに［追加（Ａ）］ボタンをクリックすると、コマンド･リスト上に順次登録される。

【０１４９】また、このコマンド･リスト上で所定の行を選択してから、［変更（Ｍ）］ボタンをクリックすることによって、当該行は変更対象となり、その設定内容がＴｙｐｅ、Ｖａｒｉａｂｌｅなどの各コンボボックスに表示される。また、このコマンド･リスト上で所定の行を選択してから、［削除（Ｄ）］ボタンをクリックすることによって、当該行をコマンド･リストから削除することができる。

【０１５０】そして、このダイアログ上で分岐ボックスの詳細の指定を完了すると、［閉じる（Ｃ）］ボタンをクリックすることによって、指定内容がＲＣＯＤＥで記述された処理ルーチンとして登録されるとともに、ダイアログは閉じる。

【０１５１】なお、ユーザは、図１１や図１２に示すよ

うなダイアログを使用して各ボックスの詳細を自ら設定する必要は必ずしもない。オーサリング･ツールが用意するＧＵＩ画面上の部品ウィンドウ（例えば図６を参照のこと）には、あらかじめ詳細な指定を完了した動作ボックスや分岐ボックスなどが部品化して用意されているので、ユーザは、所望の部品ボックスを部品ウィンドウから編集ウィンドウにドラッグ･アンド･ドロップすることにより、そのまま当該部品をプログラムの一部としてそのまま使用することができる。

【０１５２】また、オーサリング･ツールが提供するＧＵＩ画面（図６を参照のこと）で、あるボックス上でマウスの右ボタンをクリックすると「コンテクスト･メニュー」（図示しない）が出現する。このコンテクスト･メニューからは、［ボックスの削除］や［接続の削除］を行うことができる。

【０１５３】但し、ＧＵＩ画面上で、ボックス間を接続する特定の線を指定して、その削除を直接指示することはできず、線の削除には必ずコンテクスト･メニューを使用しなければならない。分岐ボックスの場合、どの接続を削除するのかを選択するためのリスト･ボックスを含んだダイアログ（図示しない）が出現するようになっている。ユーザは、該リスト中から削除したい接続を選んだ［削除］ボタンをクリックすればよい。

【０１５４】上述したようなオーサリング･ツールのＧＵＩ画面を用いて、ユーザは、例えばロボット１がサッカー･ゲームを行うようなシナリオすなわち動作手順を記述したプログラムを作成することができる。このようなシナリオは、例えば、図１３に示すようなボックス構成で記述される。

【０１５５】図１３に示す例では、動作ボックス［初期設定］において指定された内容でロボット１の初期設定を行った後、分岐ボックス［モード分岐］で指定された＊条件との比較結果に応じて、ロボット１は［転倒回復］、［探索モード］、［追跡モード］のうち１つの動作モードを実行するようになっている。

【０１５６】ここで、［転倒回復］、［探索モード］、［追跡モード］の各々はグループ･ボックスとして記述されている。前述したように、編集ウィンドウ上でグループ･ボックスをダブルクリックした場合には、そのグループが別の編集ウィンドウとして開くようになっている。図１４には、このうち［探索モード］をダブルクリックして出現する、探索モードの編集ウィンドウを図解している。また、図１５には、［追跡モード］をダブルクリックして出現する、追跡モードの編集ウィンドウを図解している。

【０１５７】本実施例では、ロボット１の処理動作は、ＲＣＯＤＥと呼ばれる、アセンブラのようなニーモニック･コードによって記述されるということは既に述べた。ここで、ＲＣＯＤＥの詳細について説明しておく。

【０１５８】ＲＣＯＤＥとは、簡単なコマンドでロボット１を制御するために策定されたプログラム言語であり、ＩＦやＧＯなどの基本的な制御構造も備えているので、ロボット制御用最低水準スクリプト言語としても使用することができる。

【０１５９】ＲＣＯＤＥでは、すべての大文字の単語は定数名などのために予約されており、ユーザが用意する変数名などに使用することができない。

【０１６０】ＲＣＯＤＥは、制御、ジャンプ、分岐、代入、同期、関数など、実行すべき行動を指定するための演算子（operator）を備えている。以下に、ＲＣＯＤＥ演算子の一覧を示しておく。

【０１６１】

【数１】

《制御》

EDIT　編集モードに移行する（シリアル・ラインを通してのコードの転送）。

RUN　実行を開始する。

END　編集モードを終了する／実行を終了する。

LOAD:<filename>　メモリ･スティック上のコードをメモリ上に読み出す

SAVE:<filename>　メモリ上のコードをメモリ･スティック上に書き込む

《ジャンプ》

:<label>　ラベル定義。<label>は１以上の数値（０は次の行を表す）。

GO:<lebel>　ラベルにジャンプする。

《分岐》

IF:<op>:<var1>:<const>|<var2>:<then_label>[:<else_label>]

変数１<var1>と定数<const>若しくは変数２<var2>を比較し、分岐する。

<op>は、"="、"<>"、"<"、"<="、">"、">="の６種類がある。

SWITCH:<var>

変数の値をコンテクスト値として設定する。

CSET:<op>:<var1>|<const1>:<var2>|<const2>:<var3>|<const3>

値１と値２を<op>演算子で比較して、その結果が真ならば、値３をコンテクスト値として設定する。

25 26

CSET命令が連続して現れた場合、真になったCSET命令以降のCSET命令はNo Operation扱いとなる（縦積み命令）。

CASE:<const>:<RCODE command>

コンテクスト値が<const>と等しければ<RCODE command>を実行する。

《代入》

LET:<var1>:<const>|<var2>　　<var1> ← <const>|<var2>

変数への値の代入

SET:<var1>:<const>|<var2>　　<var1> ← <const>|<var2>

変数によっては特殊機能が働く。（単なる代入ではない）

GET:<var1>

<var1>の内容をコンソールに表示（デバッグ用）

《演算》

ADD:<var1>:<const>|<var2>　　<var1>←<var1>+<const>|<var2>　加算

SUB:<var1>:<const>|<var2>　<var1>←<var1>−<const>|<var2>　減算

MUL:<var1>:<const>|<var2>　<var1>←<var1>*<const>|<var2>　乗算

DIV:<var1>:<const>|<var2>　<var1>←<var1>/<const>|<var2>　除算

MOD:<var1>:<const>|<var2>　<var1>←<var1>%<const>|<var2>　剰余

AND:<var1>:<const>|<var2>　<var1>←<var1>&<const>|<var2>　論理積

IOR:<var1>:<const>|<var2>　<var1>←<var1>|<const>|<var2>　論理和

XOR:<var1>:<const>|<var2>　<var1>←<var1>^<const>|<var2>　排他的論理和

RND:<var1>:<from>:<to>　<var1>←範囲 <from>～<to> の一様乱数値

《同期》

WAIT　　直前に実行した動作の終了（再生の終了）を待つ。

WAIT:<ms>　引数にしたミリ秒だけ待つ。<ms> = 1～30000。
（注：分解能は 32[ms]）

WAIT:SOUND　音再生の終了を待つ（単独で音再生中の判定にも使える）。

WAIT:LIGHT　光再生の終了を待つ（単独で光再生中の判定にも使える）。

《関数》

CALL:<label>　　<label> のサブルーチンをコールする。

※ネストは最大１６個。パラメータ渡しは変数で代用することとする。

RETURN　サブルーチンからのリターン　　。

※スタック・ポインタは無チェックなので、ネストの整合性に注意する必要がある。

【０１６２】ＲＣＯＤＥのコマンドのうち、ロボット１の「動き」を制御するコマンドは以下の５つの基本形に大別される。すなわち、

【０１６３】（１）POSE　ある姿勢（モーションの開始姿勢）をとらせる。

（２）MOVE　あるモーションをさせる。OMNE/OMTE/OMLE2_S/OMSE に対応。

（３）PLAY　あるモーションを再生させる。OAllReplay/OHeadReplay/OLegsReplay/OTailReplay に相当する。

（４）STOP　動作の停止（現在行っているモーションを終了してから）。

（５）QUIT　動作の緊急停止（直ちに）。

SET:ROBOT:POWER:1

SET:ROBOT:POWER:0

GET:ROBOT

GET:ROBOT:<var>

SET:ROBOT:<var>:<value>

＊【０１６４】MOVEとPLAYは、組み込み動作を行うか、又は、メモリ・スティック上の"*.mtn"ファイルに定義されている動作を行うかの違いだけで、「ある動作をさせる」という点では同じ機能であると言える。MOVE の動作は組み込みなので、カスタマイズできないが、細かいパラメータ指定等を行うことができる。これに対し、PLAYの動作は"*.mtn"ファイルの再生なので、パラメータ指定はできないが、差し替え可能である。

【０１６５】以下に、ＲＣＯＤＥコマンドの一覧を示しておく。

【０１６６】

＊　【数２】

電源ＯＮ

電源ＯＦＦ

全変数ダンプ

変数 取得

変数 設定

```
POSE:ROBOT:<motion_name>          motionの開始ポーズへ
POSE:LEGS:<motion_name>            motionの開始ポーズへ
POSE:HEAD:<motion_name>            motionの開始ポーズへ
POSE:TAIL:<motion_name>            motionの開始ポーズへ
PLAY:ROBOT:<motion_name>          motion再生
PLAY:LEGS:<motion_name>            motion再生
PLAY:HEAD:<motion_name>            motion再生
PLAY:TAIL:<motion_name>            motion再生
PLAY:SOUND:<sound_name>:<volume>    音再生
              volume = 0～100
PLAY:LIGHT:<pattern_name>:<times>    光再生（ＬＥＤパターン）
              times = 0～16 (0:Default 16:Loop)
MOVE:ROBOT:<motion_name>           OMLE組み込み動作再生（転倒復帰など）
MOVE:LEGS:WALK:<style1>:<dir>:<times>      歩行
MOVE:LEGS:STEP:<style1>:<dir>:<times>      歩行（StepWalk）
MOVE:LEGS:KICK:<style2>:<deg>              蹴り
              style1 = 0～10,12(R-Turn),13(L-Turn)
              style2 = 14(R-Kick),15(L-Kick)
              dir    = 1～6
              times  = 0～9999
              deg    = -90～90
MOVE:HEAD:HOME                              頭部ホーム・ポジション
MOVE:HEAD:ABS:<tilt>:<pan>:<roll>:<time>    頭部 絶対位置移動
MOVE:HEAD:REL:<tilt>:<pan>:<roll>:<time>    頭部 相対位置移動
MOVE:HEAD:C-TRACKING                        頭部カラー・トラッキング
MOVE:HEAD:C-TRACKING:<time>                頭部カラー・トラッキング
            tilt = -180～180
            pan  = -180～180
            roll = -180～180
            time = 0～99999 [ms]
MOVE:TAIL:HOME                              尾部ホーム・ポジション
MOVE:TAIL:ABS:<tilt>:<pan>:<time>          尾部絶対位置移動
MOVE:TAIL:SWING:<tilt>:<pan>:<time>        尾部スイング
            tilt = -90～90
            pan  = -90～90
            time = 0～99999 [ms]
STOP:ROBOT                                  全体通常停止
STOP:LEGS                                  四肢通常停止
STOP:HEAD                                  頭部通常停止
STOP:TAIL                                  尾部通常停止
STOP:SOUND                                  音再生停止
STOP:LIGHT                                  光再生停止
QUIT:ROBOT                                  全体緊急停止
QUIT:LEGS                                    四肢緊急停止
QUIT:HEAD                                    頭部緊急停止
QUIT:TAIL                                    尾部緊急停止
QUIT:SOUND    （STOP:SOUNDと等価）          音再生停止
QUIT:LIGHT    （STOP:LIGHTと等価）          光再生停止
```

【０１６７】また、ＲＣＯＤＥは、ロボット１の各部に対して動作・制御値を指示したり、各部の状態を取得する（センサ出力値の読み出しなど）ために、システム変数を定義している。以下に、ＲＣＯＤＥのシステム変数

の一覧を示しておく。 ＊【数３】

【０１６８】 ＊

Power	電源　0:OFF 1:ON
Head_tilt	頭部第１関節角度[°]
Head_pan	頭部第２関節角度[°]
Head_roll	頭部第３関節角度[°]
Head_mouth	頭部顎関節角度[°]
Tail_1	尾部第１関節角度[°]
Tail_2	尾部第２関節角度[°]
Leg_RF_1	右前脚第１関節角度[°]
Leg_RF_2	右前脚第２関節角度[°]
Leg_RF_3	右前脚第３関節角度[°]
Leg_LF_1	左前脚第１関節角度[°]
Leg_LF_2	左前脚第２関節角度[°]
Leg_LF_3	左前脚第３関節角度[°]
Leg_RR_1	右後脚第１関節角度[°]
Leg_RR_2	右後脚第２関節角度[°]
Leg_RR_3	右後脚第３関節角度[°]
Leg_LR_1	左後脚第１関節角度[°]
Leg_LR_2	左後脚第２関節角度[°]
Leg_LR_3	左後脚第３関節角度[°]
Head_sw	頭部圧力センサ［10^-3Pa]
Distance	頭部障害物センサ[mm]
Leg_RF_sw	右前脚肉球センサ[On:-1 Off:0]
Leg_LF_sw	左前脚肉球センサ[On:-1 Off:0]
Leg_RR_sw	右後脚肉球センサ[On:-1 Off:0]
Leg_LR_sw	左後脚肉球センサ[On:-1 Off:0]
Gsensor_status	Ｇセンサ・ステータス　１６ビット･フラグ
Gsensor_roll	Ｇセンサ・ロール角[°]
Gsensor_pitch	Ｇセンサ・ピッチ角[°]
Gsensor_yaw	Ｇセンサ・ヨー角[°]
Cdt_npixel	色センサ画素数[Pixels]
Psd_status	障害物センサ状況
Psd_range	障害物までの距離
Touch_head	頭部タッチ・センサ
Touch_head_time	頭部タッチ･センサ押下時間
Touch_head_press	頭部タッチ･センサ平均圧力　[単位不明]
Touch_RF	右前脚肉球センサ　ON→OFF／OFF→ON検出
Touch_LF	左前脚肉球センサ　ON→OFF／OFF→ON検出
Touch_RR	右後脚肉球センサ　ON→OFF／OFF→ON検出
Touch_LR	左後脚肉球センサ　ON→OFF／OFF→ON検出
Tone_num	トーン検出　音番号
Tone_level	音の大きさ
Tone_dir	検出方向
Melody_id	メロディ検出　メロディＩＤ
Melody_num	トーン数（１～３）
Tone1_num	Tone1音番号
Tone1_level	Tone1音の大きさ
Tone1_dir	Tone1検出方向

Tone2_num	Tone2音番号
Tone2_level	Tone2音の大きさ
Tone2_dir	Tone2検出方向
Tone3_num	Tone3音番号
Tone3_level	Tone3音の大きさ
Tone3_dir	Tone3検出方向
Sound_status	サウンド検出　ステータス
Sound_num	音番号
Sound_level	音の大きさ
Sound_dir	検出方向
Sound_busy	１：サウンド再生中　０：サウンド無し
Light_busy	１：Light再生中　０：Light無し

【０１６９】上記のシステム変数のうちGsensor_status は１６ビットのフラグで構成されるが、各ビットの意味は以下の通りである。

【０１７０】
【表１】

ビット	OMGsensor 状態	説　明
０ｘ０００１	FALL_DOWN_FRONT	（前方）転倒
０ｘ０００１	FALL_DOWN_RIGHT	（右側）転倒
０ｘ０００１	FALL_DOWN_LEFT	（左側）転倒
０ｘ０００１	FALL_DOWN_REAR	（後方）転倒
０ｘ０００２	HOLD_UP	抱き上げ
０ｘ０００４	HOLD_DOWN	抱き下げ
０ｘ０４００	GET_UP	転倒復帰
０ｘ０８００	JOINT_DANGER	関節挟み込み
０ｘ１０００	JOINT_GAIN_ENABLED	ゲインＯＮ
０ｘ２０００	JOINT_GAIN_DISABLED	ゲインＯＦＦ

【０１７１】また、システム変数Sound_statusは、以下のような値によってロボット１への音声入力状態を表すようになっている。

【０１７２】
【表２】

Sound_status	状　態	説　明
0	omtonePEAK	ピークのある音
1	omtoneNOPEAK	ピークのない音
2	omtoneNOPOWER	音量が小さい
3	omtoneCALC	計算中
4	omtoneOVERFLOW	オーバーフロー

【０１７３】また、ＲＣＯＤＥは、ロボット１上の各部位に配備されたセンサの出力データのログをとる（メモリ上の所定領域に一時記憶する）サービスを提供する。すなわち、ＲＣＯＤＥコマンド"LOG:START"でロギングの開始を指示するとともに、ＲＣＯＤＥコマンド"LOG:END"でロギングの終了を指示することができる。また、変数"Log_n"によって記憶したログ件数を表す。

【０１７４】さらに、ＲＣＯＤＥコマンド"LOG:PCFILE:<file>"によって、指定したファイル名でホスト・システム１００上のファイルにログを保存することができる。同様に、ＲＣＯＤＥコマンド"LOG:MSFILE:<file>"によって、指定したファイル名でメモリ・スティック上のファイルにログを保存することができる。

【０１７５】また、各センサの出力データを、メモリやファイルに保存することなく、逐次、プリンタで直接印刷出力するようにしてもよい。この場合、ＲＣＯＤＥコマンド"LOG:PRINT: <s>:<t>:<i>"で出力形式を指定することができる。但し、ｓはセンサ番号を（［表３］を参照のこと）、ｔは値の種類を（［表４］を参照のこと）、ｉはインデックス（０～Log_n-1）を、それぞれ示すものとする。また、ＲＣＯＤＥコマンド"LOG:PRINT:CRLF"で改行を、ＲＣＯＤＥコマンド"LOG:PRINT:EOF"でファイル・クローズを、それぞれ指示することができる。

【０１７６】
【表３】

s	センサ	説　明
0	Head Tilt	PRM:/r0/c0-Joint:j0
1	Head Pan	PRM:/ro/o0/c1-Joint:j1
2	Head Roll	PRM:/r0/o0/c1/c2-Joint:j2
3	Mouth	PRM:/ro/o0/c1/c2/c3-Joint:j3
4	Tail 1	PRM:/r1/c0-Joint:j0
5	Tail2	PRM:/r1/c1-Joint:j1
6	Leg FR 1	PRM:/r2/c0-Joint:j0
7	Leg FR 2	PRM:/r2/c0/c1-Joint:j1
8	Leg FR 3	PRM:/r2/c0/c1/c2-Joint:j2
9	Leg FL 1	PRM:/r3/c0-Joint:j0
10	Leg FL 2	PRM:/r3/c0/o1-Joint:j1
11	Leg FL 3	PRM:/r3/c0/c1/o2-Joint:j2
12	Leg RR 1	PRM:/r4/o0-Joint:j0
13	Leg RR 2	PRM:/r4/c0/o1-Jont:j1
14	Leg RR 3	PRM:/r4/c0/o1/o2-Joint:j2
15	Leg RL 1	PRM:/r5/o0-Joint:j0
16	Leg RL 2	PRM:/r5/o0/o1-Joint:j1
17	Leg RL 3	PRM:/r5/o0/c1/o2-Joint:j2
18	Head Press	PRM:/r0/c0/c1/c2/c3-Sensor:t3
19	Distance	PRM:/r0/c0/c1/c2/p3-Sensor:p3
20	Acc 1	PRM:/acc0-Sensor:a0
21	Acc 2	PRM:/acc1-Sensor:a1
22	Acc 3	PRM:/acc2-Sensor:a2
23	Gyro 1	PRM:/gyro0-Sensor:g0
24	Gyro 2	PRM:/gyro1-Sensor:g1
25	Gyro 3	PRM:/gyro2-Sensor:g2
26	Thermo 1	PRM:/thermo0-Sensor:th0
27	Thermo 2	PRM:/thermo1-Sensor:th1
28	Button FR	PRM:/r2/c0/o1/o2/c3-Sensor:sw3
29	Button FL	PRM:/r3/c0/c1/o2/o3-Sensor:sw3
30	Button RR	PRM:/r4/c0/o1/o2/c3-Sensor:sw3
31	Button RL	PRM:/r5/c0/c1/o2/c3-Sensor:sw3

【０１７７】　　＊　＊【表４】

t	定数名	実際のメンバー
0	VALUE	OSensorValue.value
1	SIGNAL	OSensorValue.signal
2	PWM	OJointValue.pwmDuty

【０１７８】既に説明したように、本実施例に係るオーサリング・ツールを用いてロボット１の動作制御用のシナリオを作成・編集すると、編集したシナリオを実現するためのＲＣＯＤＥに自動変換するようになっている。例えば、図１３に示すようなＧＵＩ画面上で生成されたメイン・ルーチンを持つサッカー・ゲーム用シナリオの動作制御プログラムは、以下に示すようなＲＣＯＤＥを用いて以下のように記述される。

【０１７９】

【数４】

```
SETV:ROBOT:POWER:1
SET:Trace:1
SET:mode:0 // 0:ｻｰﾁﾓｰﾄﾞ 1:追っ掛けﾓｰﾄﾞ
SET:head:0 // ｻｰﾁﾓｰﾄﾞで頭をｸﾞﾙｸﾞﾙするためのｶｳﾝﾀ
SET:lost:0 // ﾎﾞｰﾙを見失った回数
:100 // MainLoop
SET:stat:Gsensor_status
AND:stat:1
IF:=:stat:1:9000 //ずっこけた？
IF:=:mode:0:1000 //サーチモードへ
IF:=:mode:1:2000 //追っ掛けモードへ
GO:100
:1000 // mode:0 ｻｰﾁﾓｰﾄﾞ
SET:mode:0
MOVE:LEGS:STEP:RIGHT_TURN:0:4 //右回りしながらﾎﾞｰﾙを捜す
SWITCH:head //頭をｸﾞﾙｸﾞﾙまわす処理
CASE:0:MOVE:HEAD:ABS:-15:0:0:500
CASE:1:MOVE:HEAD:ABS:-15:-40:0:500
CASE:2:MOVE:HEAD:ABS:-15:-80:0:500
CASE:3:MOVE:HEAD:ABS:-45:-40:0:500
CASE:4:MOVE:HEAD:ABS:-45:0:0:500
```

35

```
CASE:5:MOVE:HEAD:ABS:-45:40:0:500
CASE:6:MOVE:HEAD:ABS:-45:80:0:500
CASE:7:MOVE:HEAD:ABS:-15:40:0:500
CASE:8:MOVE:HEAD:ABS:-15:0:0:500
ADD:head:1 // head の ｲﾝｸﾘﾒﾝﾄ
MOD:head:9
WAIT
IF:<:Cdt_npixel:32:100 // ﾎﾞｰﾙ見えていれば 追っ掛けﾓｰﾄﾞ へ
:2000 // 追っ掛けﾓｰﾄﾞ
SET:mode:1
IF:<:Cdt_npixel:32:1000    //ﾎﾞｰﾙ見えているか？
MOVE:HEAD:C-TRACKING:100   // 色追跡開始
IF:>:Head_tilt:-58:2300    // 頭の角度で距離推定
IF:>:Head_pan:0:2210:2220  // 近ければキック！
:2210
MOVE:HEAD:HOME // 自分のアゴを蹴らないように (^^;)
 ちょと不自然
MOVE:LEGS:KICK:LEFT_KICK:0
MOVE:LEGS:STEP:SLOW:FORWARD:1
GO:2900
:2220
MOVE:HEAD:HOME // 自分のアゴを蹴らないように (^^;)
 ちょと不自然
MOVE:LEGS:KICK:RIGHT_KICK:0
MOVE:LEGS:STEP:SLOW:FORWARD:1
GO:2900
:2300 // 頭の角度により、ボールに近づく
CSET:>:Head_pan:60:1
CSET:>:Head_pan:45:2
```

36

```
＊CSET:>:Head_pan:15:3
 CSET:<:Head_pan:-60:4
 CSET:<:Head_pan:-45:5
 CSET:<:Head_pan:-15:6
 CSET:=:0:0:0
 CASE:0:MOVE:LEGS:STEP:SLOW:FORWARD:4
 CASE:1:MOVE:LEGS:STEP:SLOW:FORWARD:4
 CASE:2:MOVE:LEGS:STEP:LEFT_TURN:0:4
 CASE:3:MOVE:LEGS:STEP:SLOW:LEFT:4
 CASE:4:MOVE:LEGS:STEP:SLOW:LEFTFORWARD:4
 CASE:5:MOVE:LEGS:STEP:RIGHT_TURN:0:4
 CASE:6:MOVE:LEGS:STEP:SLOW:RIGHT:4
 CASE:7:MOVE:LEGS:STEP:SLOW:RIGHTFORWARD:4
 :2900
 WAITGO：１００
 ：９０００ ／／ 転倒回復
 QUIT:ROBOT
 MOVE:ROBOT:ReactiveGU
 WAIT
 GO:100
```

【０１８０】また、図５を参照しながら説明したように、本実施例に係るオーサリング･システムでは、例えばＣ言語のような高級言語を用いてスクリプト形式で動作制御プログラムを作成･編集することができる。上記の［数４］で示したＲＣＯＤＥプログラムと等価な機能及び動作手順を実現するスクリプトを以下に示しておく。但し、スクリプトはＣ言語の文法に則って記述されているものとする。

【０１８１】

【数５】

```
//
//    スクリプト形式で記述されたサッカー･ゲーム
//
//#define ROBOT_NEUTRAL      ( 0 )      //      ニュートラル
//#define ROBOT_FOUND          ( 1 )      //      ボール見つかった
//#define ROBOT_LOST           ( 2 )      //      ボール見つからない
//#define ROBOT_FALL           ( 99 )     //      ROBOT転倒
//      ピンク
//#define PinkBall             ( 32 )
//      システム変数
extern int      Head_tilt, Head_roll;
extern int      Head_pan;
extern int      Cdt_npixel;
extern int      Gsensor_status;
//      ボールを見つけたか見失ったか
int bBall = 0, bBoil;
//      現在の状態
int nSoccer = 0;
int nLost = 0;      //      ボールを見失った時のステップ数
// 関数の前方宣言
```

37 38

```
int SoccerDog(int);
void NeutralROBOT();
void FindBall();
void SearchBall();
void RecoveryGU();
void NearBall();
void KickBall();
//    メイン
void main( void ){

    //    Soccerフラグ
    int bSoccerFlag;
    int dummy = 0, true = 1;

    bSoccerFlag = dummy = true;

    //    電源ON
    PowerOn();

    //    メインループ
    while( bSoccerFlag ){
        //    サッカードッグ
        bSoccerFlag = SoccerDog(1);
    }

    //    電源OFF
    PowerOff();

}
//    サッカー開始
int SoccerDog(int dummy ){

    //    現在の状態で処理を分ける
    switch( nSoccer ){
    case 0: //ROBOT_NEUTRAL:      //    ニュートラル状態
        NeutralROBOT();
        break;
    case 1: //ROBOT_FOUND:       //    ボールを見つけた
        FindBall();
        break;
    case 2: //ROBOT_LOST:        //    ボールを見失った
        SearchBall();
        break;
    case 99: //ROBOT_FALL:              //    アイボ転倒
        RecoveryGU();       //    転倒復帰処理
    }

    //    続行
    return 1; //FALSE;
}
```

39 40

```
//      ボールを見つけ、近づく処理
void FindBall( void ){
      int nHead;
      nHead = Head_tilt;

      //      カラートラッキング
      RcodeColorTracking(100);

      //      頭の角度でボールの距離を測定
      if( nHead > -58 ){
            //      ボールを検出
            if (Cdt_npixel < 32){
                  //      ボールを見失った
                  nSoccer = 2; //ROBOT_LOST;
            }
            else{
                  //      ボールに近づく
                  NearBall();
            }
      }
      else{
            //      ボールを蹴る
            KickBall();
      }

}
//      ボールに近づく
void NearBall( void ){

      //      首の角度でボールの位置を設定する
      if( Head_pan == 0 ){
            RcodeStepWalk(0, 1, 4); //StepWalk(FORWARD, 4);
            RcodeWait();
            return;
      }
      if( Head_pan > 60 ){
            RcodeStepWalk(0, 1, 4); //      StepWalk(FORWARD, 4);
            RcodeWait();
            return;
      }
      if( Head_pan > 45 ){
            RcodeStepWalk(0, 13, 4); //      StepWalk(LEFT_TURN, 4);
            RcodeWait();
            return;
      }
      if( Head_pan > 15 ){
            RcodeStepWalk(0, 5, 4); //      StepWalk(LEFT, 4);
            RcodeWait();
            return;
      }
```

```
	if( Head_pan < -60 ){
		RcodeStepWalk(0, 3, 4); //	StepWalk(LEFTFORWARD, 4);
		RcodeWait();
		return;
	}
	if( Head_pan < -45 ){
		RcodeStepWalk(0, 1, 4); //	StepWalk(RIGHT_TURN, 4);
		RcodeWait();
		return;
	}
	if( Head_pan < -15 ){
		RcodeStepWalk(0, 4, 4); //	StepWalk(RIGHT, 4);
		RcodeWait();
		return;
	}

	//	ボールを検出する
	if (Cdt_npixel < 32){
		nSoccer = 2; //ROBOT_LOST:
		return;
	}
}
//	ボールを蹴る
void KickBall( void ){

	//	頭の位置調整
	RcodeHead(0, 0, 0, 0);

	//	ボールが頭の左右どちら側にあるかで蹴る足を決める
	if( Head_pan > 0 ){
		RcodeKick(15, 0 );	//	左足
	}else{
		RcodeKick(14, 0 );	//	右足
	}

	//	歩行に移行するために1ステップ踏む
	RcodeStepWalk(0, 0, 1); //	StepWalk(FORWARD, 1);
}

//	ボールを探す処理
void SearcBall( void ){

	//	頭を動かす
	switch( nLost ){
	case 0:
		RcodeHead(-15, 0, 0, 500 );
		break;
	case 1:
		RcodeHead(-15, -40, 0, 500 );
		break;
```

```
        case 2:
                RcodeHead(-15, -80, 0, 500 );
                break;
        case 3:
                RcodeHead(-45, -40, 0, 500 );
                break;
        case 4:
                RcodeHead(-45, 0, 0, 500 );
                break;
        case 5:
                RcodeHead(-45, 40, 0, 500 );
                break;
        case 6:
                RcodeHead(-45, 80, 0, 500 );
                break;
        case 7:
                RcodeHead(-15, 40, 0, 500 );
                break;
        case 8:
                RcodeHead(-15, 0, 0, 500 );
                break;

        }

        //      ステップ数の設定
        nLost = (nLost + 1) % 9;

        //      右側に歩く
        RcodeStepWalk(12, 1, 4); //StepWalk(RIGHT_TURN, 4);

        //      色検出
        if (Cdt_npixel < 32){
                nSoccer = 2; //ROBOT_LOST;
        }
        else{
                //      nLostの初期化
                nLost = 0;
                //      ROBOTに近づく処理
                nSoccer = 1; //ROBOT_FOUND;
        }
}
//
//      ニュートラル時の処理
void NeutralROBOT( void ){

        //      転倒検出
        if (Gsensor_status & 1){
                //      転倒
                nSoccer = 99; //ROBOT_FALL;
                return;
```

```
        }

        //      転倒はしていない。ボール検出
        if (Cdt_npixel < 32){
            //      ボールを見失った
            nSoccer = 2; //ROBOT_LOST;
        }
        else{
            //      ボールを見つけた
            nSoccer = 1; //ROBOT_FOUND;
        }

    }
    //      転倒復帰
    void RecoveryGU( void ){

        //      転倒復帰成功？
        RcodePlay("ReactiveGU");
        RcodeWait();

        //      現在の状態をニュートラルにする
        nSoccer = 0; //ROBOT_NEUTRAL;
    }
```

【０１８２】本実施例に係るオーサリング・システムは、ユーザがテキスト・エディタなどを用いて作成・編集したスクリプトを、ＲＣＯＤＥ形式のプログラムに自動変換することができる（前述）。したがって、［数５］に示したスクリプトは、本実施例に係るオーサリング・ツールを用いて、［数４］に示したようなＲＣＯＤＥ形式のプログラム・コードに自動変換される。

【０１８３】例えば、ＲＣＯＤＥを用いて動作処理プログラムを編集するホスト・システム１００と、動作処理プログラムを実行するロボット１とが、ｂｌｕｅｔｏｏｔｈや．１１Ｂのような無線データ通信（あるいは、その他の無線又は有線の通信手段）を介して接続されている場合、ホスト・システム１は１行すなわち１ステップ単位でＲＣＯＤＥプログラムを送信し、ロボット１側ではインタープリタがこれを逐次解釈し実行することができる。また、コマンド転送時に、ホスト・システム１側は送信データを暗号化してもよい。

【０１８４】本実施例に係るオーサリング・ツールは、ＲＣＯＤＥ形式で記述されたスクリプトの編集作業、並びに、ロボット１側との協働的動作によるリアルタイムのデバッグ作業のために、編集ウィンドウを用意している。図６に示した編集ウィンドウと区別するために、この編集ウィンドウのことを、以下では、「スクリプト編集ウィンドウ」と呼ぶことにする。

【０１８５】図１６には、スクリプト編集ウィンドウの構成を図解している。

【０１８６】スクリプト編集ウィンドウは、７個のコンボ・ボックスを備えており、階層メニュー形式でコマンドを指定することができる。同図に示す例では、編集領域の第１行目のコマンドを指定する様子が示されている。

【０１８７】コンボ・ボックスの下にある［Ｃｏｍｍａｎｄ］ボタンをクリックすると、指定したコマンドをロボット１側に転送するようになっている。転送方式は、ｂｌｕｅｔｏｏｔｈや．１１Ｂなどの無線データ通信を基本とするが、その他の通信方式であってもよい。また、ここで行われるコマンド転送は、後述の［Ｅｘｅｃｕｔｅ］とは非同期に動作する。

【０１８８】また、［Ｃｏｍｍａｎｄ］ボタンの左横にある［↓］ボタンをクリックすると、指定したコマンド文字列が、下方の編集領域に貼り付けられる。

【０１８９】［Ｃｏｍｍａｎｄ］ボタンの下方に配設された編集領域上では、キー入力によりスクリプトを直接編集することができる。

【０１９０】また、編集領域のさらに下方には、［Ｏｐｅｎ］、［Ｓａｖｅ］、［Ｓｅｎｄ］、［Ｅｘｅｃｕｔｅ］、及び、［Ｓｔｏｐ］という各ボタンが配設されている。

【０１９１】［Ｏｐｅｎ］ボタンのマウス操作により、ホスト・システム１００のローカル・ディスクなどから所望のスクリプト・ファイルを選択して、編集領域に表示することができる。

【０１９２】［Ｓａｖｅ］ボタンのマウス操作により、編集領域に表示されているスクリプトを、ファイルに保

存することができる。

【０１９３】［Ｓｅｎｄ］ボタンをクリックすることにより、編集領域に表示されているスクリプトをロボット１側に転送することができる。転送方式は、ｂｌｕｅｔｏｏｔｈや．１１Ｂなどの無線データ通信を基本とするが、その他の通信方式であってもよい。但し、スクリプトは、転送されるだけで、ロボット１上でそのまま実行される訳ではない。なお、［Ｓｅｎｄ］処理の終了時には、コンソール･ウィンドウ（図示しない）で"ＥＮＤ"という行が表示される。

【０１９４】［Ｅｘｅｃｕｔｅ］ボタンをクリックすることによって、ロボット１側に送信したスクリプトの実行開始を指示することができる。

【０１９５】［Ｓｔｏｐ］ボタンをクリックすることによって、ロボット１において実行中のスクリプトの強制停止を指示することができる。

【０１９６】Ｃ．ロボットによる動作プログラムの実行／デバッグ

上述したように、スクリプト編集ウィンドウ上で［Ｃｏｍｍａｎｄ］ボタンをクリックすることによって、該当する１行のコマンドをロボット１側に転送することができる。また、［Ｓｅｎｄ］ボタンを押すことによって、編集領域に開かれているスクリプトをロボット１側に転送することができる。このようなコマンド転送時には、コマンド･データを一旦暗号化してから送出することが好ましい。

【０１９７】他方のロボット１側では、ＲＣＯＤＥなどで記述された動作制御プログラムの実行及びデバッグ環境として、インタープリタ／デバッガと、ミドルウェアと、ドライバと、オペレーティング・システム（ＯＳ）とを備えている（図５を参照のこと）。

【０１９８】インタープリタは、ＲＣＯＤＥ形式で記述されたプログラムを１行ずつ読み込んで解釈して実行する高水準言語プログラムである。また、デバッグ時などにおいて、ホスト・コンピュータ１００側から暗号化された形式でＲＣＯＤＥプログラムが送信される場合には、インタープリタは、これを一旦復号化してから解釈・実行を行う。

【０１９９】デバッガは、ＲＣＯＤＥプログラム中の誤り（バグ）を発見して、修正する作業を支援するプログラムである。すなわち、プログラムを指定した行で実行を止めたり、そのときのメモリや変数の内容を参照することができる。

【０２００】ミドルウェアは、ロボット１における歩行その他の脚式動作などの制御や、カメラ１５からの入力画像の認識処理や、マイク１６からの音声入力の認識処理などを行う処理エンジンである。ミドルウェアは、ドライバを介して各関節アクチュエータやその他のハードウェアの操作を行うことができる。

【０２０１】ミドルウェアは、ロボット１の脚式動作制御において、目標とされる姿勢や目標とされる動作に遷移するための情報（ここでは、仮に「姿勢遷移情報」と呼ぶことにする）を生成する。すなわち、インタープリタなど上位プログラムから供給される行動指令情報に基づいて、ロボット１の現在の姿勢又は動作から次の姿勢又は動作（目標とされる姿勢又は目標とされる動作）に遷移させるための姿勢遷移情報を生成する。現在の姿勢から次に遷移可能な姿勢は、胴体や手や脚の形状、重量、各部の結合状態のようなロボット１の物理的形状や、各関節の曲がる方向や角度などの装置構成によって決定される。ミドルウェアは、このような装置構成を考慮して姿勢遷移情報を生成する。

【０２０２】本実施例では、ミドルウェアは、ロボット１が遷移可能な姿勢及び遷移する際の動作を、例えば有向グラフの形式であらかじめ登録しており、有向グラフに従って上位プログラムからの行動指令を姿勢遷移情報に変換して、ドライバに対する具体的な動作指示を発行するようになっている。

【０２０３】一般に、ロボット１は、現在の姿勢によっては、ＲＣＯＤＥで記述された指令内容に従って直接遷移できない場合がある。すなわち、ロボット１の姿勢は、現在の姿勢から直接遷移可能な姿勢と、直接には遷移できなくてある動作や姿勢を経由してなら可能となる姿勢とに分類される。

【０２０４】例えば、ロボット１が図１に示すように４足歩行型のロボット装置である場合、手足を大きく投げ出して寝転んでいる状態から伏せた状態へ直接移動することはできるが、立った姿勢へ直接遷移することはできず、一旦手足を胴体近くに引き寄せて伏せた姿勢を経由して、それから立ち上がるという少なくとも２段階の動作が必要である。

【０２０５】また、このように２段階の動作を経由しなければ安全に（すなわちロボット１が転倒することなく）実行できないような姿勢遷移も存在する。例えば、図１に示すような４足歩行型のロボット１は、立っている姿勢で両前足を挙げて「万歳」をしようとすると転倒してしまう場合がある。あるいは、現在の姿勢が寝転び姿勢（寝姿勢）にあるときに、指令の内容として座り姿勢しかできないような「足をバタバタさせる」といった内容が送られてきたとき、そのまま足をバタバタさせる命令を各関節駆動部に発行してしまうと、寝姿勢から座り姿勢への遷移と足をバタバタさせる動作とが実行されてしまうことになる。この結果、ロボット１はバランスを崩して転倒してしまうであろう。

【０２０６】本実施例では、ミドルウェアは、ロボット１がとり得る姿勢及び動作が登録されるとともに、各姿勢を遷移させる動作で結んだ有向グラフを保持している。そして、現在の姿勢から目標とされる（すなわち指示された）姿勢又は動作までの経路を、この有向グラフ上で探索して、その探索結果に基づいて、現在の姿勢か

ら目標とされる姿勢又は動作へと遷移させるような命令を生成するようになっている。

【０２０７】図１７には、本実施例において使用される有向グラフの具体例を図解している。有向グラフは、ロボット１がとり得る姿勢を示すノードと、遷移可能な２つのノード（姿勢）間を結ぶ有向アーク（動作アーク）とで構成される。図１７に示す例では、ノードすなわち姿勢として、「ねそべる」、「ふせる」、「すわる」、「たつ」、「あるく」という５種類が含まれる。また、有向アークは、ある姿勢から他の姿勢へ遷移する動作アークと、他の姿勢からある姿勢へ戻る動作アークと、１つのノード内で動作が完結する自己動作アークとがある。また、ある姿勢から他の姿勢へ遷移する動作アークが複数あることや、ある姿勢について複数の自己動作アークがあることもある。

【０２０８】例えば、ロボット１の現在の姿勢が「ふせる」であるときに、インタープリタに対して「すわる」ことを指示するＲＣＯＤＥコマンドが投入された場合、図１７に示すように、「ふせる」と「すわる」の各ノード間には有向アークが存在するので直接遷移が可能である。したがって、ミドルウェアは、インタープリタから渡される指示に従って、ドライバに対して駆動指令を発行すればよい。

【０２０９】これに対し、現在の姿勢が「ふせる」であるときに、インタープリタに対して「あるく」ことを指示するＲＣＯＤＥコマンドが投入された場合、図１７に示すように、「ふせる」というノードから「あるく」というノードへ直接結ぶ有向アークは存在しない。したがって、ミドルウェアは、有向グラフ上でノード「ふせる」からノード「あるく」へ向かう経路探索して、ノード「たつ」を経由してノード「ふせる」からノード「あるく」に遷移するという姿勢遷移計画を立て、該計画に基づいてドライバに対して駆動指示を発行する。

【０２１０】以上を略言すれば、本実施例によれば、ミドルウェアがロボット１の姿勢・動作遷移特性を管理しており、ＲＣＯＤＥコマンド・レベルでの論理的な姿勢・動作指示とハードウェア上の姿勢・動作遷移特性との差異を吸収できるメカニズムを提供している。

【０２１１】したがって、一般ユーザは、オーサリング・ツールを用いてＧＵＩ画面上でロボット１の動作シナリオを編集したり、あるいはスクリプト形式で動作制御プログラムを記述するような場合に、現実のロボット１における姿勢・動作遷移特性のような詳細なハードウェア情報を気にしなくて済む。また、ミドルウェア・レベルにおいてＲＣＯＤＥコマンド・レベルでの論理的な姿勢・動作指示とハードウェア上の姿勢・動作遷移特性との差異を吸収するので、ロボットの姿勢・動作遷移特性に反していることはＲＣＯＤＥプログラムのデバッグの対象とはならない。

【０２１２】［追補］以上、特定の実施例を参照しながら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が該実施例の修正や代用を成し得ることは自明である。

【０２１３】本実施例では、イヌを模した４足歩行を行うペット型ロボットを例に挙げて本発明に係るオーサリング・システムについて詳解したが、本発明の要旨はこれに限定されない。例えば、ヒューマノイド・ロボットのような２足の脚式移動ロボットや、あるいは脚式以外の移動型ロボットに対しても、同様に本発明を適用することができることを充分理解されたい。

【０２１４】あるいは、ロボットのような物理的な機械装置ではなく、コンピュータ・グラフィックスにより生成されるキャラクタを用いたアニメーションの動作シーケンスの作成・編集のために、本発明に係るオーサリング・システムを適用することも可能である。

【０２１５】要するに、例示という形態で本発明を開示してきたのであり、限定的に解釈されるべきではない。本発明の要旨を判断するためには、冒頭に記載した特許請求の範囲の欄を参酌すべきである。

【０２１６】

【発明の効果】以上詳記したように、本発明によれば、ロボットの所定の動作パターンを記述する一連のコマンド／データを作成することができる、優れたオーサリング・システム及び方法を提供することができる。

【０２１７】また、本発明によれば、ロボットの動作状態を規定する部品の集合を用いて動作パターンを作成することができる、優れたオーサリング・システム及び方法を提供することができる。

【０２１８】また、本発明によれば、各部品をコンピュータ・ディスプレイ上に配置して動作パターンを作成することができる、優れたオーサリング・システム及び方法を提供することができる。

【図面の簡単な説明】

【図１】本発明を実施に供される四肢による脚式歩行を行う移動ロボット１の外観構成を示した図である。

【図２】移動ロボット１の電気・制御系統の構成図を模式的に示した図である。

【図３】制御部２０の構成をさらに詳細に示した図である。

【図４】ホスト・コンピュータ１００のハードウェア構成例を模式的に示した図である。

【図５】本実施例に係るオーサリング・システムの全体構成を模式的に示した図である。

【図６】本実施例に係るオーサリング・ツールが提供する、ユーザがロボット１の動作シナリオを作成・編集するためのＧＵＩ画面の構成を示した図である。

【図７】メニュー項目「ファイル」に関するプルダウン・メニューを示した図である。

【図８】メニュー項目「グループ」に関するプルダウン・メニューを示した図である。

【図９】編集ウィンドウ上に動作ボックス［Ａ］及び分岐ボックス［Ｂ］が各１個配置されている様子を示した図である。

【図１０】図９に示す編集ウィンドウ上で、［ＳＴＡＲＴ］ボックスと動作ボックス［Ａ］、並びに、動作ボックス［Ａ］と分岐ボックス［Ｂ］を接続した様子を示した図である。

【図１１】動作ボックスの詳細を指定するためのダイアログを示した図である。

【図１２】分岐ボックスの詳細を指定するためのダイアログを示した図である。

【図１３】ロボット１がサッカー・ゲームを行うためのシナリオを、オーサリング・ツールのＧＵＩ画面上で作成した様子を示した図である。

【図１４】図１３に示すサッカー・ゲームのシナリオ中のグループ・ボックス［探索モード］を別の編集ウィンドウとして開いた様子を示した図である。

【図１５】図１３に示すサッカー・ゲームのシナリオ中のグループ・ボックス［追跡モード］を別の編集ウィンドウとして開いた様子を示した図である。

【図１６】本実施例に係るオーサリング・ツールが用意するスクリプト編集ウィンドウの構成を示した図である。

【図１７】ロボット１側のＲＣＯＤＥプログラム実行環境において保持される、ロボット１の姿勢及び動作の遷移を規定する有向グラフを示した図である。

【符号の説明】

１…移動ロボット

２…胴体部ユニット

＊３…頭部ユニット

４…尻尾

６Ａ～６Ｄ…脚部ユニット

７…首関節

８…尻尾関節

９Ａ～９Ｄ…大腿部ユニット

１０Ａ～１０Ｄ…脛部ユニット

１１Ａ～１１Ｄ…股関節

１２Ａ～１２Ｄ…膝関節

１５…ＣＣＤカメラ

１６…マイクロフォン

１７…スピーカ

１８…タッチセンサ

２０…制御部

２１…ＣＰＵ

２２…ＲＡＭ

２３…ＲＯＭ

２４…不揮発メモリ

２５…インターフェース

２６…無線通信インターフェース

２７…ネットワーク・インターフェース・カード

２８…バス

２９…キーボード

４０…入出力部

５０…駆動部

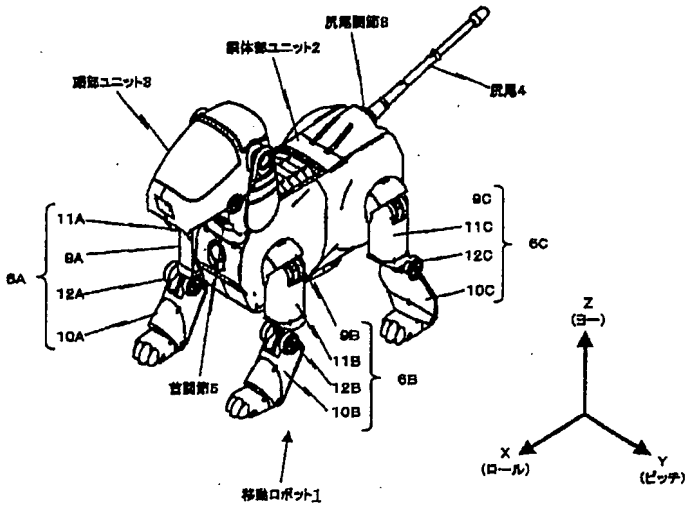
５１…モータ

５２…エンコーダ

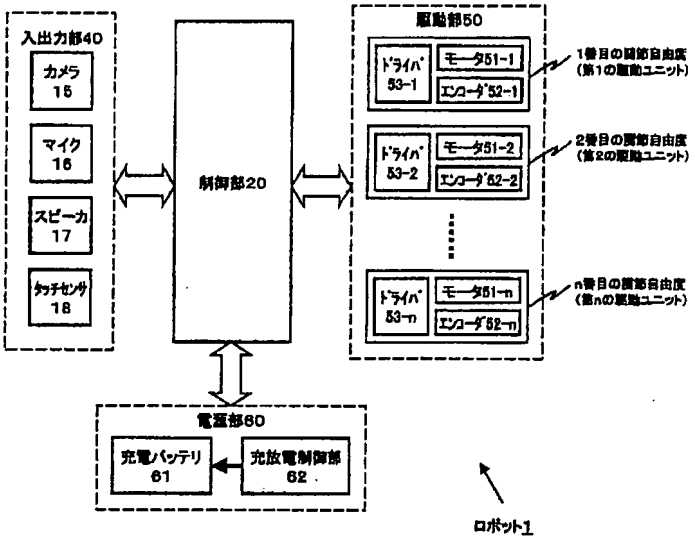
５３…ドライバ

＊

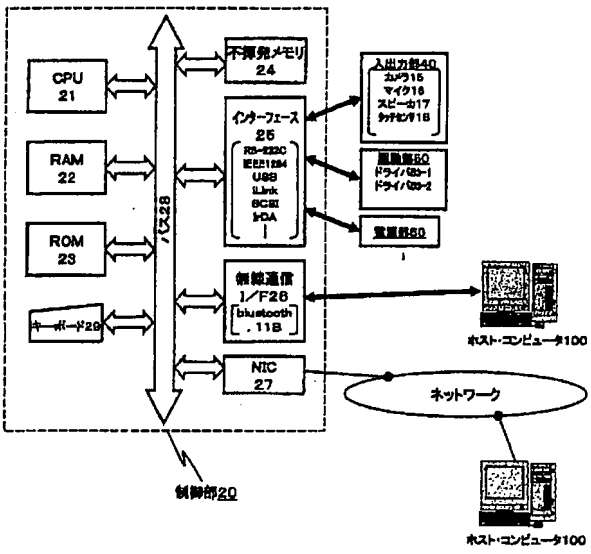
【図１】



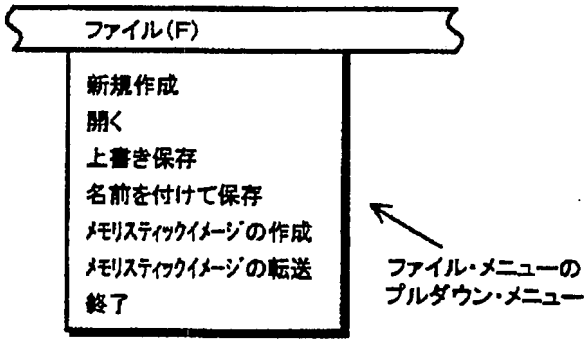
【図２】


入出力部40
カメラ 15
マイク 16
スピーカ 17
タッチセンサ 18
制御部20
駆動部50
ドライバ 53-1
モータ51-1
エンコーダ52-1
1番目の関節自由度（第1の駆動ユニット）
ドライバ 53-2
モータ51-2
エンコーダ52-2
2番目の関節自由度（第2の駆動ユニット）
ドライバ 53-n
モータ51-n
エンコーダ52-n
n番目の関節自由度（第nの駆動ユニット）
電源部60
充電バッテリ 61
充放電制御部 62
ロボット1

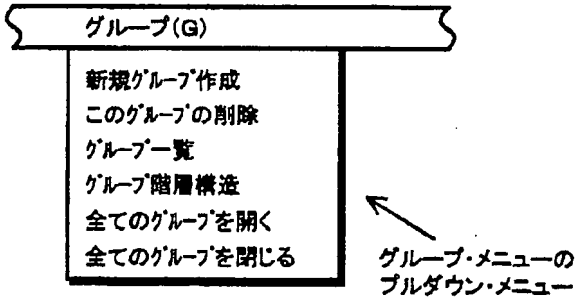
【図３】


CPU 21
RAM 22
ROM 23
キーボード29
バス28
不揮発メモリ 24
インターフェース 25
RS-232C
IEEE1394
USB
USB
SCSI
IrDA
入出力部40
駆動部50
電源部60
無線通信 I/F26
bluetooth
802.11b
NIC 27
ネットワーク
ホスト・コンピュータ100
ホスト・コンピュータ100
制御部20

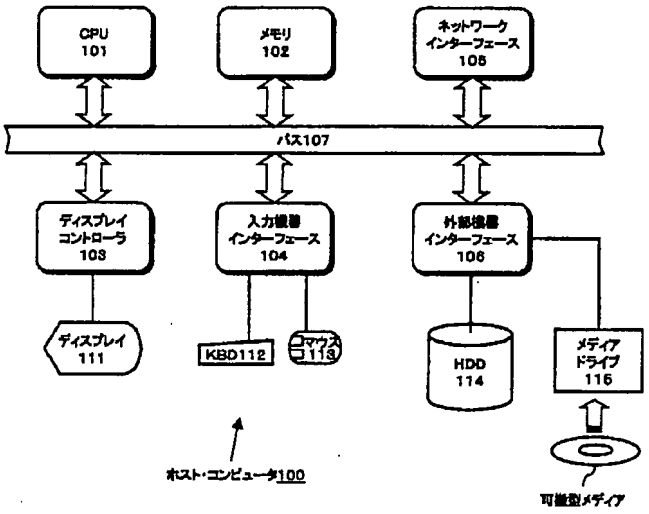
【図７】


ファイル(F)
新規作成
開く
上書き保存
名前を付けて保存
メモリスティックイメージの作成
メモリスティックイメージの転送
終了
ファイル・メニューのプルダウン・メニュー

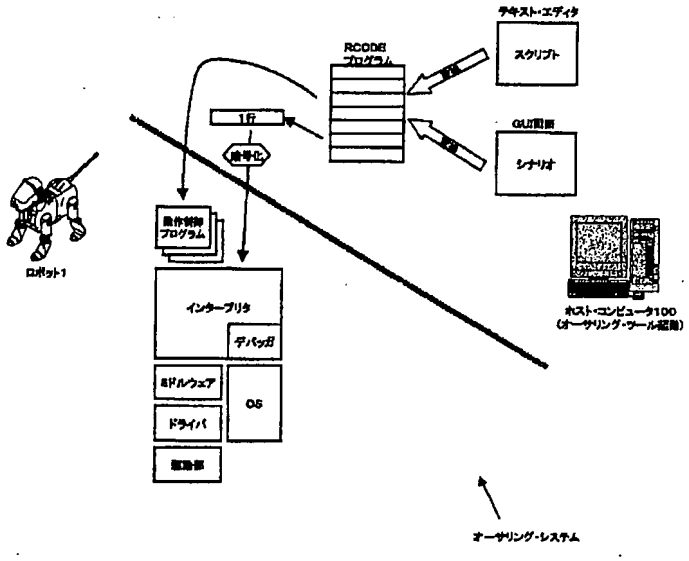
【図８】


グループ(G)
新規グループ作成
このグループの削除
グループ一覧
グループ階層構造
全てのグループを開く
全てのグループを閉じる
グループ・メニューのプルダウン・メニュー

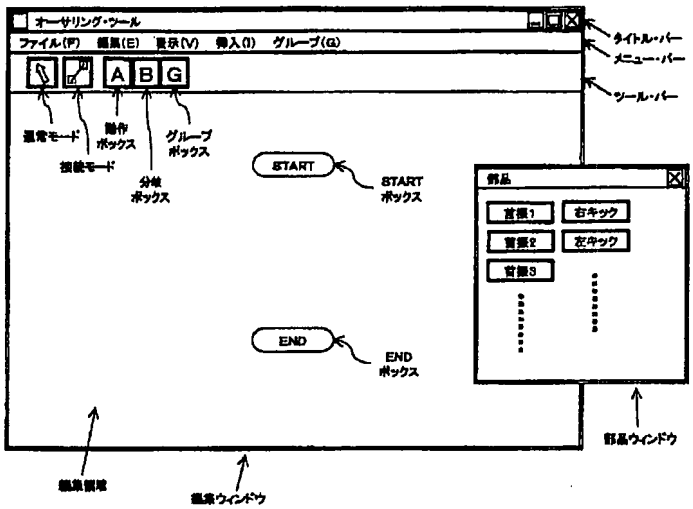
【図４】


CPU
101
メモリ
102
ネットワーク
インターフェース
105
バス107
ディスプレイ
コントローラ
103
入力機器
インターフェース
104
外部機器
インターフェース
106
ディスプレイ
111
KBD112
マウス
113
HDD
114
メディア
ドライブ
116
ホスト・コンピュータ100
可搬型メディア

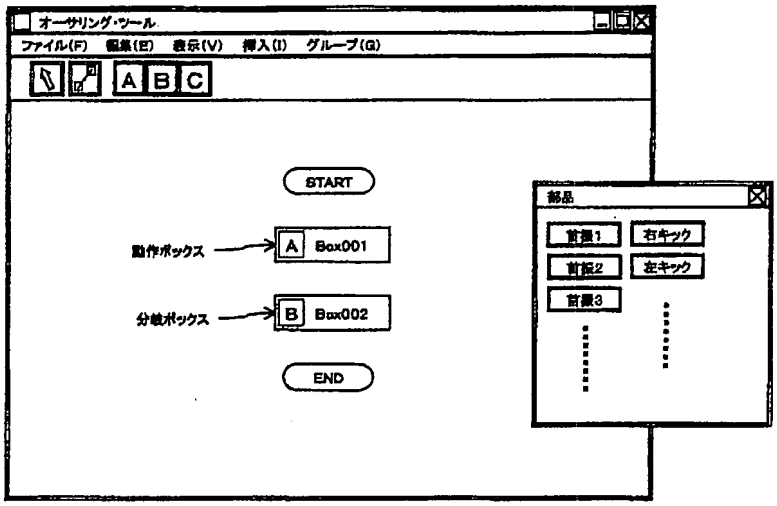
【図５】


テキスト・エディタ
スクリプト
RCODE
プログラム
GUI画面
シナリオ
1行
逐次化
動作制御
プログラム
ロボット1
インタープリタ
デバッガ
ミドルウェア
OS
ドライバ
ホスト・コンピュータ100
(オーサリング・ツール起動)
オーサリング・システム

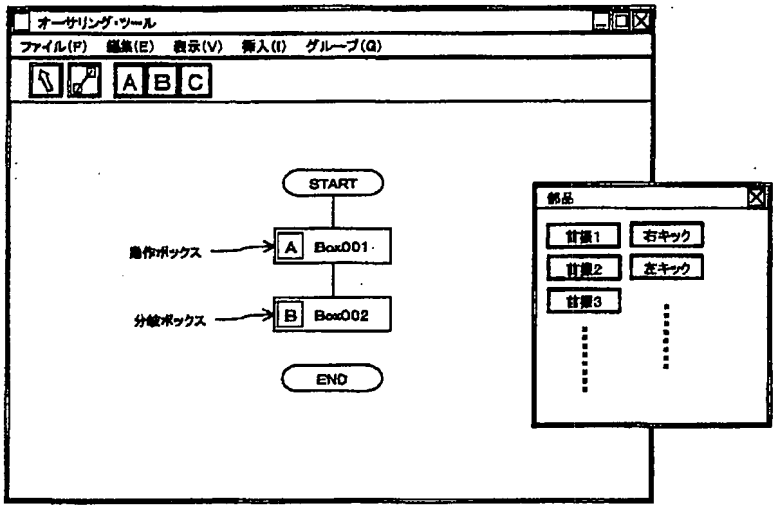
【図６】


オーサリング・ツール
ファイル(F) 編集(E) 表示(V) 挿入(I) グループ(G)
A B G
タイトル・バー
メニュー・バー
ツール・バー
通常モード
接続モード
動作ボックス
分岐ボックス
グループボックス
START
START ボックス
END
END ボックス
部品
首振1
右キック
首振2
左キック
首振3
部品ウィンドウ
編集領域
編集ウィンドウ

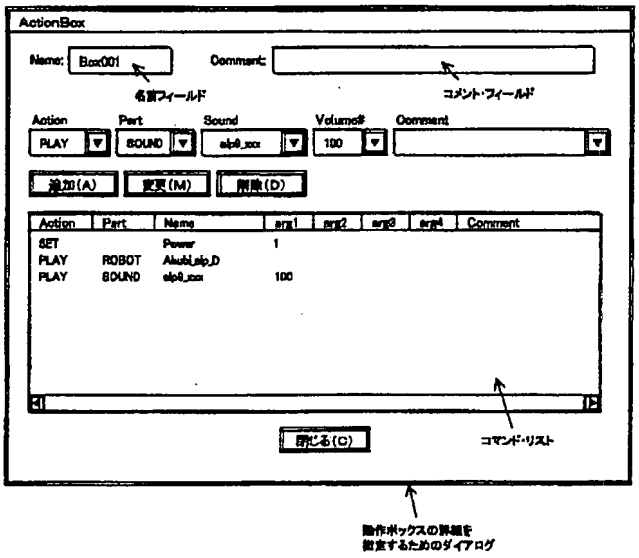
【図９】


オーサリング・ツール
ファイル(F) 編集(E) 表示(V) 挿入(I) グループ(G)
A B C
START
動作ボックス
A Box001
分岐ボックス
B Box002
END
部品
首振1
右キック
首振2
左キック
首振3

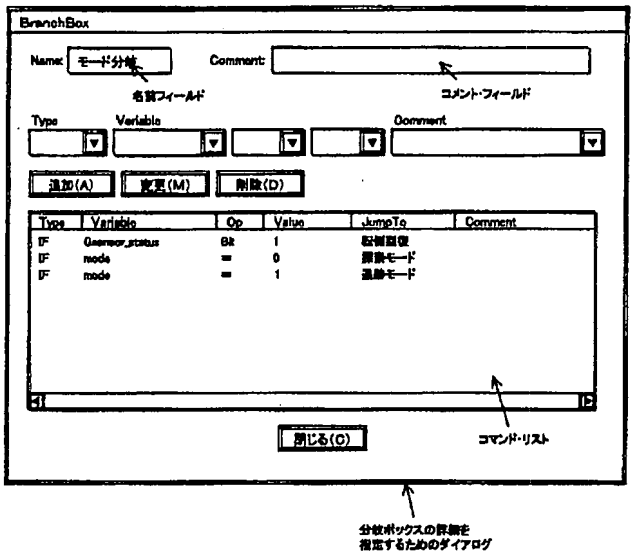
【図１０】


オーサリング・ツール
ファイル(F)　編集(E)　表示(V)　挿入(I)　グループ(G)
A B C
START
動作ボックス
A Box001
分岐ボックス
B Box002
END
部品
右キック
左キック

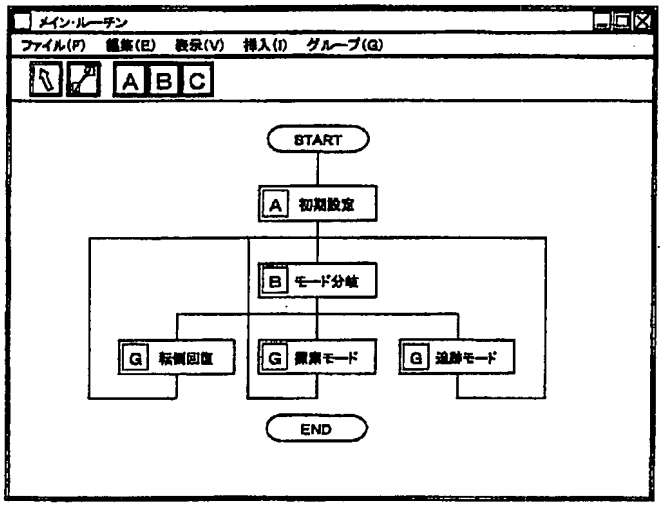
【図１１】


ActionBox
Name: Box001
Comment:
名前フィールド
コメント・フィールド
Action
Part
Sound
Volume#
Comment
PLAY
SOUND
100
追加(A)
変更(M)
削除(D)
Action
Part
Name
arg1
arg2
arg3
arg4
Comment
SET
Power
1
PLAY
ROBOT
PLAY
SOUND
100
閉じる(C)
コマンド・リスト
動作ボックスの詳細を
設定するためのダイアログ

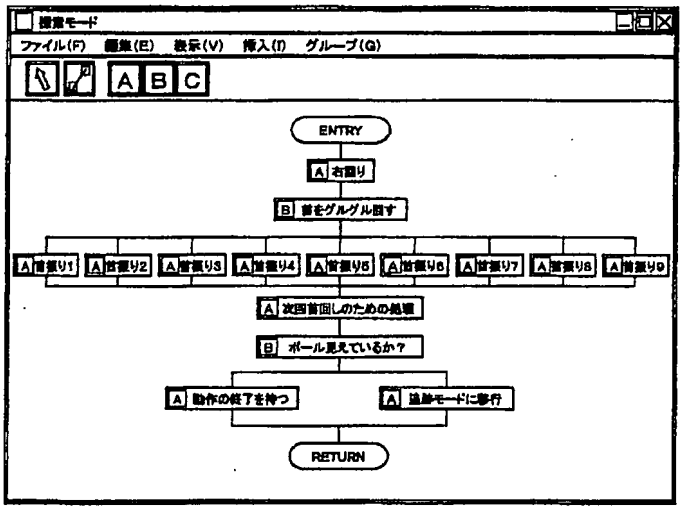
【図１２】


BranchBox
Name:
モード分岐
Comment:
名前フィールド
コメント・フィールド
Type
Variable
Comment
追加(A)
変更(M)
削除(D)
Type
Variable
Op
Value
JumpTo
Comment
IF
Sensor_status
==
1
転倒回復
IF
mode
==
0
探索モード
IF
mode
==
1
追跡モード
閉じる(C)
コマンド・リスト
分岐ボックスの詳細を
指定するためのダイアログ

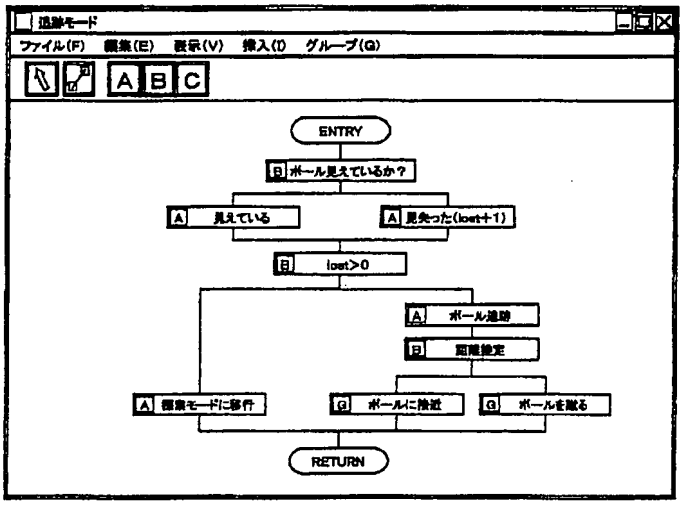
【図１３】


メイン・ルーチン
ファイル(F)
編集(E)
表示(V)
挿入(I)
グループ(G)
A
B
C
START
A 初期設定
B モード分岐
G 転倒回復
G 探索モード
G 追跡モード
END

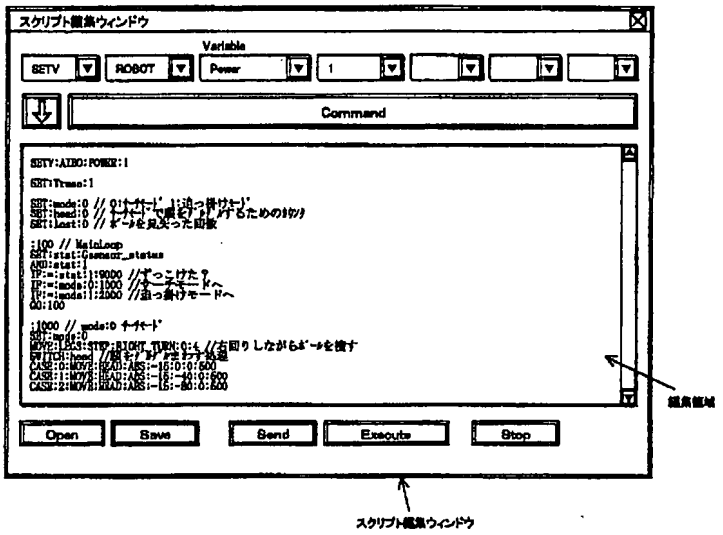
【図１４】


探索モード
ファイル(F) 編集(E) 表示(V) 挿入(I) グループ(G)
A B C
ENTRY
A お辞儀
B 首をグルグル回す
A首振り1
A首振り2
A首振り3
A首振り4
A首振り5
A首振り6
A首振り7
A首振り8
A首振り9
A 次回首回しのための処理
B ボール見えているか？
A 動作の終了を待つ
A 追跡モードに移行
RETURN

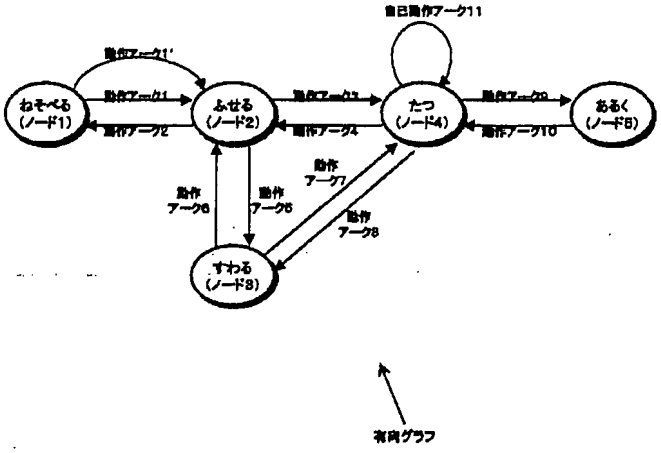
【図１５】


追跡モード
ファイル(F) 編集(E) 表示(V) 挿入(I) グループ(G)
A B C
ENTRY
B ボール見えているか？
A 見えている
A 見失った(lost+1)
B lost>0
A ボール追跡
B 距離推定
A 探索モードに移行
G ボールに接近
G ボールを蹴る
RETURN

【図１６】



【図１７】



フロントページの続き

Ｆターム(参考) 3F059 AA00 BA02 BB06 BC07 BC09
CA05 CA06 DA02 DA05 DA08
DB04 DB08 DB09 DC01 DC04
DD01 DD04 DD08 DD11 DD18
DE05 FA03 FA05 FB01 FB05
FC02 FC07 FC13 FC14
3F060 AA00 CA14 CA26 GA05 GA13
GB25 GD13 GD15 HA02 HA32
HA35